Exploration de micro-posts d'actualité : représentation, structuration et description

Olivier Gracianne

UNIVERSITÉ D'ORLÉANS

École Doctorale Mathématiques, Informatique, Physique Théorique et Ingénierie des Systèmes

THÈSE présentée par :

Olivier GRACIANNE

soutenue le : **13 décembre 2023**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Informatique**

Exploration de micro-posts d'actualité : représentation, structuration et description

THÈSE DIRIGÉE PAR :

Mme DAO Thi-Bich-Hanh	Maître de conférences HDR, Université d'Orléans
Mme HALFTERMEYER Anaïs	Maître de conférences, Université d'Orléans

---

JURY :

M. ANTOINE Jean-Yves	Professeur des universités, Université de Tours	**Président**
M. NAACKE Hubert	Maître de conférences HDR, Sorbonne Université Paris 6	**Rapporteur**
M. ROCHE Mathieu	Directeur de recherche, CIRAD	**Rapporteur**
Mme DAO Thi-Bich-Hanh	Maître de conférences HDR, Université d'Orléans	**Examinateur**
M. VALETTE Mathieu	Professeur des universités, INALCO	**Examinateur**
Mme VRAIN Christel	Professeur des universités, Université d'Orléans	**Examinateur**
Mme HALFTERMEYER Anaïs	Maître de conférences, Université d'Orléans	**Membre invité**
M. GARRIGA Laurent	Directeur de l'innovation, Atos	**Membre invité**

Remerciements

Je souhaite remercier en premier lieu Mesdames Thi-Bich-Hanh Dao et Anaïs Halftermeyer, Maîtresses de conférences à l'Université d'Orléans, pour m'avoir encadré et soutenu tout au long de cette thèse. Même dans les périodes les plus difficiles de la poursuite de nos travaux, leur patience, leurs idées et leur aide m'ont permis de continuer d'avancer et de mener à bien ce projet.

J'aimerais également remercier Monsieur Laurent Garriga, directeur technique et innovation à Atos, référent industriel de cette thèse, pour avoir apporté son regard et maintenu un cadre idéal pour la poursuite de nos travaux.

Je souhaite également remercier Messieurs Hubert Naacke et Mathieu Roche, pour avoir accepté d'être les rapporteurs du présent manuscrit. Je remercie Madame Christel Vrain ainsi que Messieurs Jean-Yves Antoine, Lakhdar Sais et Mathieu Valette pour avoir accepté de faire partie du jury.

Je remercie l'école doctorale MIPTIS, l'Université d'Orléans, le Laboratoire d'Informatique Fondamentale d'Orléans et tout leur personnel pour m'avoir accueilli et m'avoir permis de poursuivre cette thèse dans les meilleures conditions.

Je veux également remercier Atos pour m'avoir offert la possibilité de poursuivre des travaux de thèse en partenariat avec un groupe industriel, et pour m'avoir accompagné pendant ces dernières années.

Je tiens à remercier Madame Isabelle Renard, secrétaire au LIFO, pour sa disponibilité, sa réactivité et son efficacité remarquables. Son travail nous a permis de nous concentrer sur nos travaux et nous a épargné bien des déboires.

Enfin, je souhaite remercier tous les doctorants qui ont menés leurs travaux en même que moi au sein du LIFO, et en particulier Monsieur Sofiane Elguendouze, avec qui les échanges ont inspiré une partie importante des présents travaux de recherche.

Je dédie cette thèse à ma conjointe, qui a toujours été à mes côtés et m'a toujours soutenu quand j'en avais le plus besoin. Son amour et son soutien inconditionnel m'ont porté du début à la fin de ces travaux.

À chacun des membres de ma famille, pour n'avoir jamais cessé de croire que je pouvais mener au bout cette thèse et ne jamais avoir oublié de me le dire.

À chacun de mes amis chers, pour m'avoir aider à garder l'esprit sain et à relâcher la pression quand il le fallait.

Sommaire

Introduction

Contexte scientifique

Le "micro-blogging" est une forme de communication sur le web. Elle consiste à échanger à travers des messages appelés micro-posts, ou micro-blogs, du fait de leur courte taille. Cette contrainte de volume a pour but de différencier le contenu d'une telle plate-forme d'un réseau plus classique, en favorisant les communications régulières aux messages longs.

Twitter est le plus connu des médias de ce type. Il est utilisé d'une part pour les discussions entre ses usagers mais surtout pour diffuser de l'information. Aujourd'hui, c'est une vitrine pour la communication d'une grande majorité des entreprises et de la quasi-totalité des institutions publiques. En outre, presque toute la presse traditionnelle s'est dotée de comptes et d'équipes utilisant ce réseau pour collecter et diffuser du contenu.

C'est une source de données de premier choix pour la description d'évènement d'actualité. Cela est du, en particulier, à la rapidité avec laquelle l'information y est relayée mais aussi à la diversité des sources la diffusant. La presse traditionnelle est loin d'être la seule à poster des micro-posts, ou des tweets, sur ce qui est en train de se passer. Les presque 240 millions d'utilisateurs journaliers de cette plate-forme peuvent écrire, et/ou photographier, pour communiquer sur ce qui se déroule sous leurs yeux. Les sources d'informations au sein de Twitter sont donc légion et très variées. En écoutant ce qui se dit d'un évènement sur la plate-forme, on aura accès à toute l'information "standard" qu'on peut trouver à son sujet. Elle relève de celle que l'on pourrait trouver dans des journaux mais aussi de tous les éléments de plus petite granularité qui peuvent échapper à d'autres recherches d'information.

De multiples travaux ont déjà été menés pour exploiter l'information qui circule sur ce réseau et s'en servir pour différents objectifs, allant de l'accélération de la réponse à un séisme à l'estimation de l'efficacité d'une campagne marketing. En outre, on peut voir à travers ces données la réalité du discours qui est employé par tout un chacun pour parler d'un évènement et de ses différentes manifestations.

Pour une entreprise comme Atos, le partenaire industriel de la convention encadrant ces travaux, cette forme de communication représente un enjeu critique. L'entreprise est donc active dans ce domaine de recherche pour y maintenir un haut niveau de compétence, et pouvoir développer des solutions innovantes permettant d'en tirer profit. C'est notamment le cas pour le Lab d'Olivet. Ces structures de l'entreprise sont celles qui mènent la grande majorité de ses travaux d'innovations. Celui d'Olivet est spécialisé dans les technologies liées au traitement du langage et à l'automatisation de processus. Participer à des travaux de thèse sur un sujet comme le notre permet ainsi à cette structure de renforcer son portefeuille de compétences sur l'exploitation des réseaux sociaux.

En particulier, le problème abordé dans ces travaux est la description d'actualité à

travers les données issues de plate-formes de micro-blogging.

Description d'actualités à partir de micro-posts

Qu'est-ce qu'un évènement ?

S'intéresser à l'actualité consiste concrètement à s'intéresser aux évènements qui la composent. Nous nous attachons à décrire des ensembles de données récupérées sur Twitter et ainsi, il faut définir ce que nous appelons "évènement" à travers ces corpus. Il convient donc de présenter la définition adoptée ici pour le terme évènement. Par "évènements", nous entendons des évènements du monde réel pouvant impacter l'activité humaine d'une entreprise, d'une ville, d'une région. Dans les faits, cela peut donc consister en un rassemblement sportif, un meeting de grande envergure ou un phénomène météorologique de forte magnitude.

Nous nommons donc "évènement" la cible des requêtes que nous construisons avec des mots-clefs qui permettent de les identifier de la manière la plus large possible. Par exemple pour une tempête, on pourrait bien évidemment utiliser "tempête", et l'éventuel nom de celle-ci si on le connaît au moment de formuler la requête. Mais être plus spécifique que cela à travers des a priori sur la tempête, comme en utilisant "pluie" ou "vent", biaiserait la collecte des données. Les messages collectés devant contenir au moins un des mots de la requête, nous captons ainsi tous ceux qui en contiennent un et avons de la sorte du contenu à propos de la tempête sans l'orienter sur un de ses possibles aspects.

Ces évènements sont complexes et peuvent être structurés en sous-parties, de différentes natures, profondeurs et complexités. En effet, les contenus sont diffusés par une très grande diversité d'utilisateurs et donc de points de vue. Ces derniers postent en outre leur message depuis des endroits et à des moments très variés, et chacun le fait dans son idiosyncrasie. La requête permettant de capter un jeu de données à propos de l'évènement lui garantit une consistance lexicale, du fait de la présence d'au moins un des termes de la requête. Nous faisons donc l'hypothèse que le découpage de celui-ci sur une base sémantique permet d'identifier des sous-ensembles de tweets cohérents, imageant au moins partiellement ces sous-parties. Nous appelons donc "sous-évènements" ces ensembles de tweets de granularité plus fine qui représentent les différentes facettes, ou manifestations, de l'évènement que l'on peut identifier dans les données. Ils ont vocation à être liés ontologiquement à l'évènement cible. Ils peuvent également n'avoir qu'un lien lexical avec celui-ci, dû par exemple à une homonymie ou l'utilisation métaphorique des mots de la requête. Une tempête météorologique peut par exemple porter le même nom qu'une célébrité ou avoir lieu dans la même temporalité qu'une "tempête politique".

Nous ne faisons donc pas d'hypothèses sur ce que peut contenir un corpus à propos d'un évènement mais nous attachons à décrire son contenu de la façon la plus représentative possible. Nous nous attachons à **décrire les données par les données** elles-mêmes, tentant d'identifier au plus juste ce qu'elles contiennent. Si un jeu de données à propos d'une tempête contient suffisamment de messages à propos d'un objet qui n'est pas lié à cette tempête, nous devons décrire cet objet comme une sous-partie du corpus traité. Nous considérons alors comme bruit dans nos données le contenu présent dans une proportion trop peu significative pour constituer une sous-partie à part entière.

Nature et acquisition des données traitées

L'intérêt du traitement des données issues des réseaux sociaux s'accompagne d'une difficulté de traitement inhérente à leur exploitation. Elles sont formulées en langue naturelle et ne sont soumises à aucune autre contrainte de forme qui faciliterait son traitement automatique.

La langue qui est utilisée sur Twitter est relativement standardisée pour les messages provenant d'acteurs officiels. Pour tous les autres, il s'agit de discours plus ou moins soignés dont le traitement est compliqué selon plusieurs aspects. D'abord, même si cela varie selon la langue employée, le discours se présente au travers d'une très grande diversité morphologique des mots. À cela s'ajoute également tout un ensemble d'acronymes utilisés comme des interjections dans les conversations en lignes et les fautes d'orthographe, volontaires ou involontaires, diversifiant encore les formes existantes d'un mot.

Le langage pratiqué sur Twitter compte également des éléments qui lui sont propres comme les adresses ou les hashtags, des outils qui agrandissent d'autant le lexique à la disposition d'un utilisateur de la plate-forme. En ajoutant un '#' devant un mot, celui-ci devient sur Twitter un hashtag, un technomorphème (Paveau, 2013) à l'utilité multiple. Il raccroche le message qui le contient à une tendance et/ou apporte des informations de contextes supplémentaires au message. Dans tous les cas, il augmente la visiblité de ce dernier sur la plate-forme. De plus, il peut tout à fait être intégré au corps du message comme n'importe quel autre mot, sans perdre les caractéristiques que nous venons de citer. Les hashtags sont donc autant d'éléments de lexiques supplémentaires, et modifient encore l'usage de la langue du message du fait de leur utilité sur la plate-forme. Le même constat peut être dressé pour les adresses. Un nom d'utilisateur précédé d'une '@' est-il utilisé pour interpeller l'utilisateur en question et commencer une conversation, ou simplement pour attirer son attention sur le message ? Cela illustre la difficulté de traitement de ces éléments propres à Twitter.

Pour pouvoir représenter la réalité du discours autour d'un objet discuté sur les réseaux, il faut naturellement commencer par capter ce discours. Dans notre cas, cela prend la forme du requêtage de Twitter qui nous permet d'obtenir de larges corpus de données textuelles. Le très grand nombre d'utilisateurs de la plate-forme et son accessibilité assure une bonne représentativité, aussi bien socialement que linguistiquement, de ses usagers (au moins pour les langues les plus utilisées dans le monde). Chaque utilisateur exprime un fragment de réalité d'une manière qui lui est propre. En en captant un très grand nombre, nous pouvons appréhender la ou les formes du discours partagé autour d'un évènement.

Cette contrainte d'acquisition est d'abord sujette aux limitations imposées par la plate-forme. En dehors des contraintes techniques de l'API de Twitter, le fonctionnement par mot-clef même impose des contraintes fortes sur l'acquisition des données. Il n'est possible de capter que partiellement l'information qui circule à propos d'un sujet donné en le qualifiant avec un lexique fini, puisqu'on est ainsi contraint de s'appuyer sur une connaissance a priori de ce sujet. La dynamique de la langue pratiquée sur les réseaux sociaux induit qu'un requêtage à base de mots produira nécessairement des données bruitées. Il faut pouvoir s'adapter à cette caractéristique de nos jeux de données. Les collecter en grand nombre avec des mot-clefs peu spécifiques permet de couvrir efficacement la cible de la requête.

Approche proposée

L'exploitation de la langue naturelle nécessite d'avoir recours à des processus de traitement spécialisés qui permettent d'ingérer les données venant de la plate-forme et de les traiter numériquement pour en encaisser la volumétrie. C'est à cette fin que nous avons recours à des techniques issues du monde du traitement du langage, et plus particulièrement de l'apprentissage automatique appliqué à ce domaine. Notre objectif est de traiter les données sur la base du sens exprimé par les messages, de façon à identifier les sous-parties de l'objet cible.

Nous devons pouvoir mesurer cette proximité sémantique, pour l'exploiter afin de détecter la structure formée par les éléments composant la cible. Pour ce faire, nous nous saisissons de représentations vectorielles raffinées. Ces dernières permettent de placer les documents dans un espace vectoriel continu, où la distance qui les sépare représente l'éloignement entre leur sujet. Ces vecteurs, appelés plongements, sont calculés par des modèles d'apprentissage automatique qui capturent, au moins partiellement, une sémantique distributionnelle dans les données textuelles. Nous n'approfondissons toutefois pas plus notre usage du traitement de la langue au delà de la préparation et du changement de représentation des données textuelles. Nous adoptons une chaîne de traitement, presque agnostique de la langue traitée, qui nous permet de proposer une méthodologie d'exploration et de description de ces données.

Dans un tel espace de représentation, il est possible d'identifier des zones plus "chaudes", c'est-à-dire des zones où les documents sont regroupés du fait de leur similarité. C'est à travers cela que nous proposons d'identifier les sous-ensembles de documents à décrire pour rendre compte du jeu de données complet. Dans un tel espace, il est en effet possible de découvrir ces sous-ensembles au moyen de clustering. Ce type de méthodologie permet d'identifier des structures sous-jacentes dans des jeux de données si on peut comparer les instances qui les composent au moyen d'une mesure de similarité. Les clusters ainsi ainsi découverts forment les "sous-évènements" de la cible du corpus.

Le modèle de plongement utilisé doit correspondre à nos besoins. Il doit produire des représentations permettant d'identifier la structuration sémantique des données sur lesquelles nous travaillons. Le modèle doit donc permettre d'établir une mesure de similarité entre les documents composant les corpus. Mais il ne peut pas le faire à n'importe quel prix. Les ressources de calculs qu'il nous est possible de mobiliser sont modestes. Par ailleurs, l'application finale de nos travaux sera assez spécifique pour qu'il n'existe pas d'autres jeux de données équivalents disponibles. Il faut donc que le modèle soit peu coûteux en calcul et qu'il puisse apprendre un espace de représentation robuste même en s'appuyant sur peu de données.

Finalement, la description d'un sous-ensemble des données capturées doit permettre de comprendre à quel objet sémantique il est lié. Elle doit donc être représentative de son contenu et en mettre en avant les aspects caractéristiques. Décrire un objet repose sur le constat de ses caractéristiques à travers l'utilisation de descripteurs pertinents pour les qualifier et/ou les quantifier. Dans le cadre du problème que nous traitons, cela consiste à identifier quels sont les mots, au sein des documents traités, les plus appropriés pour décrire les sous-ensembles de messages cohérents en terme de sens.

Une fois que les sous-ensembles à décrire ont été identifiés, il faut donc mettre une procédure de sélection des mots dans les documents qui les contiennent. Ce processus

doit être à même d'identifier des descripteurs de qualité pour permettre de construire des descriptions de qualité. Toutefois, la nature diffuse du vocabulaire des données textuelles rédigées en langue naturelle pose le problème suivant. Un mot sélectionné car étant adéquat pour décrire un cluster peut aussi convenir pour en décrire un autre. Il faut donc mettre en place une procédure ayant pour but de garantir que l'assignation d'un mot à la description d'un cluster soit la meilleure possible.

C'est à travers la mosaïque des descriptions de ses sous-évènements que nous proposons de décrire l'évènement cible. Il faut noter que ces descriptions sont donc contextuelles au sens où elles dépendent de l'empan temporel couvert par la requête.

Notre approche doit prouver son intérêt sur différents corpus pour montrer sa robustesse. Nous avons donc mené des expérimentations sur deux jeux de données que nous avons collectés sur Twitter, à propos d'évènements très différents. Le premier concerne la tempête Alex, qui a frappé la France en 2020, et le second est à propos de la guerre en Ukraine, qui a commencé en 2022 et est toujours en cours au moment de la rédaction du présent document. La tempête Alex a notamment été particulière du fait d'avoir frappé les Alpes-Maritimes dans le sud-est dans un premier temps, puis la Bretagne dans le nord-ouest quelques jours après. Elle a été marquée par des crues violentes pendant sa première partie, et par des vents particulièrement puissants lors de son passage sur les côtes bretonnes. Le corpus sur la guerre Russo-Ukrainienne est marqué par la surprise qu'a provoqué le conflit. Celle-ci est rapidement remplacée dans les données par les réactions de la communauté internationale, qu'elles consistent en des communications officielles ou des discussions animées entre particuliers.

Si le premier est en français et le second en anglais, leurs différences vont au delà de la langue des messages. Leur dynamique, les réactions qu'ils provoquent et la diversité de personnes qu'ils concernent ne sont pas les mêmes. Si des constatations similaires et des résultats pertinents peuvent malgré cela être obtenus avec la même approche sur ces deux jeux de données, cela constituera une première preuve robuste de son intérêt.

L'approche que nous proposons et les expérimentations que nous avons conduites avec ont fait l'objet d'une publication dans la "Conférence Nationale sur l'Intelligence Artificielle" (CNIA) de 2022 dans l'"International Conference on Tools with Artificial Intelligence" (ICTAI) de 2022.

Un objectif clef de nos travaux est de produire des descriptions intelligibles pour un humain. En effet, la description d'élément d'actualité que nous souhaitons mettre en place a vocation à être présentée à un utilisateur. Les efforts particuliers que nous avons fournis dans ce sens, notamment pour améliorer les résultats présentés à ICTAI 2022, ont fait l'objet d'une publication à ICTAI 2023.

Organisation du manuscrit

Le présent manuscrit est structuré comme suit. La partie I contient les chapitres d'état de l'art dans lesquels nous posons les notions et les domaines de recherches abordés. Nous y passons également en revue les travaux proches des nôtres sur les différents problèmes que nous abordons. Le chapitre 1 aborde les problématiques liées au traitement du langage. En particulier, il pose les définitions de nos objets d'études ainsi que les outils à mettre en oeuvre pour pouvoir les traiter. Le chapitre 2 présente les outils qu'il nous faut déployer pour pouvoir explorer nos jeux de données et y détecter les structures que nous souhaitons

décrire. Enfin, le chapitre 3 passe en revue les techniques abordant spécifiquement la tâche de description de données textuelles, pour y situer nos travaux.

La partie II est consacrée à la présentation de notre approche, de ses spécificités et des expériences que nous avons menées. Le chapitre 4 présente les outils de préparation des données, communs à tous les travaux que nous avons conduits. Le chapitre 5 correspond à notre publication dans ICTAI 2022, avec notre approche de description sur des jeux de données figés et les résultats qu'elle produit. Le chapitre 6 se penche sur l'amélioration que nous avons apportée à cette approche, dans le but de pouvoir détecter de meilleurs composants de descriptions. Finalement, le chapitre 7 présente les premiers travaux exploratoires que nous avons menés pour adapter notre approche aux données en flux.

Le dernier chapitre conclut sur ce que nous avons réalisé au cours de cette thèse et présente les pistes que nous envisageons de poursuivre à différents termes.

Publications

- Gracianne O., Halftermeyer A. et Dao T.B.H., "Des clusters de tweets aux tags de descriptions : présentation d'un évènement par la caractérisation de ses manifestations", in Conférence Nationale en Intelligence Artificielle 2022 (CNIA 2022)
- Gracianne O., Halftermeyer A. et Dao T.B.H., "Presenting an event through the description of related tweets clusters", in 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 1283-1290)
- Gracianne O., Halftermeyer A. et Dao T.B.H., "Identifying relevant descriptors for tweet sets", in 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)

Première partie

État de l'art

Chapitre 1

Traitement des données textuelles

Sommaire

Les travaux présentés dans ce manuscrit traitent de données textuelles en langue naturelle. La richesse et la complexité des données linguistiques offrent un terrain propice à l'extraction de connaissances et la compréhension des contenus textuels. Ce chapitre présente cette exploration en examinant les méthodes et les outils qui permettent d'exploiter ce type et cette modalité de données. Nous aborderons les techniques d'extraction

d'informations à partir de la langue naturelle, en présentant une façon de l'aborder. De plus, nous nous pencherons sur les avancées récentes en matière de plongement neuronal de texte, une approche qui propose de représenter les mots et les documents sous forme de vecteurs numériques. Enfin, nous discuterons des méthodes d'évaluation de ces représentations vectorielles, mesurant leur qualité et leur pertinence. Ce chapitre offre ainsi un aperçu complet de l'ensemble du processus, depuis la langue naturelle brute que nous captons, jusqu'à la création et à l'évaluation de représentations vectorielles informatives.

1.1 Données textuelles

La modalité textuelle permet de représenter sur papier les communications humaines. Elle se décline en une très grande diversité. Nous nous intéressons à la langue employée sur Twitter pour en extraire des informations particulières. Les traitements que nous employons pourraient être assez peu spécifiques au langage mais doivent considérer une co-occurrence de mots au sein d'un même document comme faisant sens. Nous donnons donc dans cette section des définitions pour éclaircir ce que sont nos objets d'étude : langue naturelle, corpus, mots et documents.

1.1.1 Langue naturelle

La langue naturelle désigne le système de communication que les humains emploient pour communiquer entre eux. Elle permet d'adopter différents niveaux de langue.

Ces différents niveaux sont utilisés pour s'exprimer sur "la toile" tout en ayant des contraintes communes liées à ce mode de communication, comme les contraintes de taille des messages ou le fait qu'un message publié en dehors d'une conversation privée soit accessible par toute personne ayant accès au Web. Notre approche propose de déduire et d'extraire des informations en s'appuyant sur les productions plutôt que de formuler clairement en amont ce que nous cherchons, du fait de la nature même de la langue naturelle. Un langage formel comme un langage informatique a pour but d'être interprété par une machine, le C ou le Python, par exemple, doivent permettre la construction de programmes. Pour cela, il doit comporter le moins d'ambiguïté possible pour éviter d'aboutir à des comportements imprévisibles. A contrario la langue naturelle, à travers ses variations, est toujours susceptible d'être ambiguë. Par ailleurs, l'ensemble des énoncés d'une langue ne seront jamais exhaustivement produits. C'est pourquoi nous prenons le parti de travailler à partir des données et non à partir d'un système a priori, un modèle lexico-syntaxique par exemple, pour extraire nos objets.

En outre, la langue naturelle se démarque sur un autre aspect. Elle est hautement dynamique et en perpétuelle évolution. Dans (Marchello-Nizia, 2018), divers types d'évolutions sont présentées. Une langue peut par exemple emprunter des éléments de lexique venant d'une autre langue et les intégrer au sien, comme le verbe "purchase" ("acheter") en anglais emprunté au vieux français "prochacier" (signifiant "se procurer"), ou tous les anglicismes existants dans le français actuel. Ces lexicalisations peuvent également avoir lieu au sein d'une même langue, avec l'apparition de nouveaux mots, comme "le vivre-ensemble". Ce dynamisme se traduit aussi au niveau de la graphie des éléments avec lesquels la langue naturelle est transcrite. Outre les variations grammaticales ou orthographiques par rapport au canon vernaculaire, volontaires ou relevant du défaut, on peut

trouver dans le vocabulaire employé dans une discussion des interjections qui servent aux locuteurs à transcrire ce qui relève du non verbal. Cette catégorie contient par exemple les acronymes comme "lol" ou "mdr", qui permettent de faire passer le rire par le texte.

Sur les réseaux sociaux, elle est d'autant plus variée. En effet de nouveaux éléments de langages comme les hashtags ("#alpesMaritimes") ou les mentions ("@nice06"), appelés technomorphèmes dans (Paveau, 2013), diversifient encore le vocabulaire à la disposition des locuteurs. Les réseaux sociaux sont par ailleurs l'endroit où l'on constate le plus rapidement les nouveaux usages de la langue ainsi que ses modes et tendances. Par exemple en anglais, l'utilisation des interjections "lol" [1] et "lmao" [2] est toujours répandue alors que celle de "rofl" [3], très courante dans les années 2000, se fait de plus en plus rare. Le français de Twitter s'est récemment vu enrichir du mot "quoicoubeh", une chimère sans sens particulier utilisée pour répondre à une phrase terminant par "quoi", dans la blague consistant à faire la même chose avec "feur".

La langue naturelle, et en particulier celle pratiquée sur le web, est ainsi beaucoup plus complexe à traiter que la langue formelle et toute description qui en est faite est amenée à devenir incomplète voire obsolète avec le temps. Son traitement automatique est donc plus difficile à mettre en oeuvre et requiert d'avoir recours à des outils et des techniques qui s'adaptent à son dynamisme. Cela motive la mise en place d'une chaîne de traitement totalement automatique, qui n'ait pas besoin de mobiliser des ressources tierces et qui se maintienne à jour.

1.1.2 Échantillon statique : corpus

À travers nos travaux, nous n'abordons pas la langue en tant que système. Celle que nous traitons s'apparente à la parole au sens de Saussure (de Saussure, 1971), elle tient de la production linguistique comme du discours, et nous l'attaquons à travers des corpus pour en identifier les spécificités. Notre but est de décrire un évènement en cours dans la réalité "extra linguistique" et dont les locuteurs, ou "twittos", traitent à travers leurs publications sur le réseau.

La composition d'un jeu de données est un élément auquel il faut prêter attention. Un corpus est selon l'*Encyclopedia Universalis* (Universalis, 2023) « un rassemblement de textes ou une collection de textes regroupés sur la base d'hypothèses de travail en vue de les interroger ». Dans (Mayaffre, 2002), Mayaffre discute de ces caractéristiques et de ce qu'un corpus permet de dire de la question de recherche pour lequel il est composé. Il le qualifie comme étant « un objet heuristique. C'est une construction arbitraire, une composition relative qui n'a de sens, de valeur et de pertinence qu'au regard des questions qu'on va lui poser, des réponses que l'on cherche, des résultats que l'on va trouver ». Ainsi, nous considérons nos corpus de tweets comme des échantillons du contenu de Twitter, des prismes par lesquels il est possible de qualifier certains aspects d'un évènement. Dans le cas des corpus finis, comme celui pour la tempête Alex, nous nous appuyons sur les dates officielles des évènements cibles pour commencer et arrêter leur constitution. Pour les corpus dynamiques, dont nous parlons plus en détail dans la prochaine section, les dates de début et de fin de collecte sont celles du lancement et de l'arrêt du système.

1. "laughing out loud"
2. "laughing my ass out"
3. "rolling on the floor laughing"

La collecte des corpus pour un cas d'usage se fait au moyen de l'API Twitter. Cette dernière est organisée en points d'accès qui permettent de récupérer des messages directement en flux ou en ensembles figés. Les messages vieux de plus de sept jours ne sont pas accessibles sur le même point d'accès. Peu importe celui qui est choisi, les tweets sont récupérés au moyen de requêtes à base de mots clefs et d'opérateurs logiques simples sur les attributs d'un tweet. Ce mode de requêtage établit un premier filtre sur le contenu des messages à traiter et doit donc être pensé en conséquence. Comme notre objectif est de décrire des évènements à partir de ce qu'il s'en dit sur la plate-forme, nous n'avons aucune connaissance sur le contenu que nous allons capter au moment d'écrire la requête. Il faut ainsi que cette dernière soit la plus générique possible par rapport à l'évènement cible, pour ne rater aucune information à son sujet en s'appuyant sur des a priori qui pourraient mener à rater certains de ses aspects.

1.1.3 Flux de données et corpus dynamique

La description d'évènement depuis du contenu venant de Twitter peut être opérée sur des ensembles statiques de tweets, par exemple en traitant des tweets publiés avant le moment de la captation des messages. Le principal avantage d'adapter ce traitement à Twitter reste toutefois de pouvoir le faire en temps réel, en traitant le contenu au sujet d'un évènement cible au fil de sa publication. 500 millions de tweets sont publiés chaque jour, en moyenne 350 000 par minute, d'où le besoin de pouvoir spécifier quels sont les éléments que l'on veut récupérer dans ce flux. On peut donc recevoir en continu de nouveaux messages et les intégrer aux corpus connus. Les traitements doivent être adaptés à cette volumétrie, en termes de rapidité d'exécution et d'espace mémoire occupé.

Un jeu de données dynamique ne peut pas être traité de la même manière que s'il est statique. Une mesure qui s'appuierait sur l'ensemble des données devrait alors être recalculée à l'arrivée de chaque nouvelle instance du jeu de données. On peut catégoriser les jeux de données dynamiques selon le domaine de définition des attributs des instances. D'un côté, on retrouve les cas où tous les attributs sont connus à l'avance. C'est par exemple celui des sites de commerce en ligne. Tout le catalogue des produits du site est connu à l'avance. Les instances sont les paniers d'achats des clients et on les reçoit au fur et à mesure que des paniers sont validés. Ainsi, si on ne peut pas savoir quels seront les produits dans un panier et dans quelle quantité ils y seront, on ne peut pas y trouver de produit inconnu (Manchanda et al., 1999). L'autre cas de figure est celui où l'ensemble des attributs des instances n'est pas connu à l'avance. C'est celui du traitement de message en flux sur Twitter. Même si on utilisait les 64 000 entrées du Petit Larousse 2023, il n'y aurait aucune garantie de connaître tous les mots du prochain message que l'on va recevoir. Par exemple dans (Tarrade et al., 2022) les auteurs proposent une méthodologie qui leur a permis d'identifier 141 changements et 251 buzz lexicaux entre mars 2012 et février 2014. Ils désignent par "changement lexical" une variation de forme ou l'apparition d'un nouveau mot dont l'utilisation se stabilise au cours du temps, et par "buzz lexical" l'emploi épisodique d'un mot qui disparaît rapidement. Dans ce genre de cas, les traitements doivent être adaptés à la volatilité des attributs car, par exemple, toutes les comparaisons qui s'appuient sur eux ne peuvent plus être effectuées, comme les mesures de distance ou de similarité (voir partie 2.1.2).

Dans tous les cas, il faut définir les unités sur lesquelles nous souhaitons baser notre

raisonnement.

1.1.4 Mot et document

Le terme "mot" désigne généralement une séquence finie de symboles formant les unités avec lesquelles sont construites les phrases. Sa définition ne fait toutefois pas consensus. Dans (Gardiner, 2016), la distinction entre "mot" et "phrase" est discutée et l'auteur propose une définition pour chacun d'eux. Cette question dépassant le cadre de nos travaux, nous utiliserons le terme "mot" pour désigner les mots qui apparaissent dans un document, à distinguer des "tokens". Ces derniers, que nous appelons aussi unités, désignent les segments de documents identifiés par leur découpage. Ces unités peuvent éventuellement comporter plusieurs mots, comme nous le verrons dans le chapitre 6.

On peut attribuer au terme "document" différentes définitions selon le domaine et le cas d'usage qui est traité. Dans sa "Théorie du document" (Buckland, 2018), Buckland explore et discute des multiples aspects du concept de document allant de la sociologie à la sémiotique. Étymologiquement, un document est une preuve, un spécimen ou un cours. On peut globalement le considérer comme une instanciation d'une partie du réel, qui peut prendre différentes formes. La délimitation qu'impose un document sur l'objet dont il traite lève également des questions sur l'information qu'il convoie, sur la construction collective de sens et la communication dans une société (Zerubavel, 1996).

Dans ces travaux, nous ne nous intéressons pas à redéfinir la notion de document. Nous traitons des messages courts publiés sur les réseaux sociaux. Un document désigne donc pour nous le corps d'un micro-post et il est accompagné par un identifiant unique, un auteur et une date de parution. Il est important de positionner la notion de contexte par rapport à celle de document. Le contexte désigne le voisinage dans lequel un mot apparaît. Il consiste donc en une séquence ordonnée de mots, elle même sous-séquence du document. Le document est donc un ensemble de contextes.

Dans le cadre de nos travaux, un document est un tweet et donc un échantillon de discours, « un discours n'[étant] pas qu'une simple suite d'énoncés posés les uns à côté des autres » (Charolles, 1995). Le document est dès sa genèse pensé comme une unité de contexte cohérente et devient avec sa construction une unité de contexte cohésive. Son sens s'exprime au travers de ses mots et il faut donc identifier le sens de ces derniers pour pouvoir accéder à celui du document. Toutefois, plusieurs documents au sein d'un corpus peuvent contenir les mêmes mots. La localité de l'unité de contexte représentée par un document permet d'attribuer un sens spécifique à ses mots. C'est ainsi que nous exploitons l'articulation entre mots et documents : les mots composent un document, ce dernier représentant une granularité de contexte dans lequel les mots ont un sens particulier. La composition de ces sens particuliers devient le sens du document. Cet aller-retour entre mots et contextes forme notre base de travail.

Exemple 1.1.1. Considérons les deux tweets suivants, tirés du même corpus.

```
@heb386nrd63h @Mediavenir tes rigolo !cette tempête est ultra
rare, plus d'une année de pluie en quelque heure ... que ce
soit en bretagne, ou dans le.centre, c'est impossible de prévoir
que sa arriverai ... la météo ne ce prevoit pas dix ans avant
```

et

`Tout de même incroyable cette météo : on alterne soleil/tempête de pluie en cinq minutes top chrono.`

Ces deux messages parlent de la météo, en utilisant des termes similaires. Toutefois, leur objectif n'est pas le même. Le premier est une réponse teintée d'agacement à un autre message quand le deuxième est une constatation étonnée. Le document constitue donc un cadre qui peut changer l'interprétation des contextes qui le composent. Leur composition est donc différente de leur somme.

La localité des mots en tant qu'unités des documents permet de capturer le sens spécifique qu'ils prennent dans un contexte donné. Par exemple, on voit dans le Tableau 1.1 que le mot `tempête` peut avoir des significations différentes selon les usages qui en sont faits. Cela montre l'intérêt de calculer la sémantique de documents à partir des mots qui les composent et la sémantique des mots à partir des contextes de leurs apparitions, ici trouvés dans les limites du document. Cela permet aussi de s'adapter aux nouveaux usages qui apparaissent avec l'utilisation de la langue naturelle. Pour pouvoir en faire autant, il faut s'intéresser à la façon d'utiliser et de désigner l'environnement d'un mot.

TABLEAU 1.1 – Exemple de localité du sens du mot tempête dans le corpus tempête Alex

mot	tweet dans lequel le mot apparaît	signification dans ce contexte
tempête	`Des inondations "historiques" dans les Alpes-Maritimes au passage de la tempête Alex - vidéos , les détails`	évènement climatique
tempête	`Tempête politique au Canada et nouveau mea culpa de Justin Trudeau`	évènement politique important
tempête	`Faire du sport écouter pnl tempête avec un casque >>>`	nom d'une chanson

1.1.5 Paradigme, contexte et co-texte

Usuellement, le contexte d'un mot désigne la situation dans laquelle il est énoncé. Il peut décrire la personne qui dit ce mot, le moment où elle le fait ou la situation dans laquelle elle se trouve. Sur Twitter, le contexte peut donc intégrer la tendance dans laquelle s'inscrit un message, son auteur, la ou les communautés auxquelles il appartient. Comme nous nous intéressons uniquement au corps d'un tweet, le contexte des mots est alors restreint à son co-texte, soit aux mots qui apparaissent autour de lui dans le message. Cela justifie l'intérêt pour les paradigmes auxquels peuvent appartenir les mots. Si on peut les identifier dans un contexte relativement restreint comme un voisinage de termes, on peut alors le faire de proche en proche jusqu'à identifier comment ce mot et son paradigme interagissent avec les autres. Par la suite, nous désignerons par "contexte" d'un mot le voisinage de mots au sein duquel il apparaît.

Puisque nous nous intéressons à l'usage in situ des mots, nous cherchons à capturer les paradigmes qui existent au sein de nos jeux de données et de les exploi-

ter pour qualifier la sémantique des documents et des mots. Nous appelons paradigme un ensemble de mots intervertibles par commutation dans la phrase où ils se trouvent sans que cela rende cette phrase incompréhensible ou n'en change la compréhension, au sens de Saussure (de Saussure, 1971). Par exemple, en reprenant l'exemple `Des inondations "historiques" dans les Alpes-Maritimes au passage de la tempête Alex - vidéos, les détails` du Tableau 1.1, `inondations` pourraient appartenir au paradigme des noms de phénomènes climatiques faisant intervenir des précipitations et comprenant par exemple `crues, intempéries, pluies`. Comme nous traitons d'échantillons spécifiques de la langue, nous ne nous intéressons qu'aux paradigmes qui existent dans ces échantillons.

Le traitement automatique de nos corpus requiert donc de représenter les mots et les documents d'une manière qui permettent de saisir l'articulation entre mots, contextes et documents.

1.2 Extraire des informations de données textuelles

Comme nous cherchons à décrire des évènements depuis des données venant de Twitter, nous cherchons à extraire les informations pertinentes au sujet de cet évènement au sein de ces corpus. Nos travaux se positionnent donc au plus près de la Recherche d'Information (RI), une tâche spécifique de la fouille de texte.

1.2.1 La fouille de texte : Historique et tâches principales

Les applications de la fouille de texte sont nombreuses. Elles vont de l'indexation de documents à la recherche et l'extraction d'information pour des usages tels que le traitement de données scientifiques en bioinformatique ou encore la détection de spams pour le grand public. Nous présentons ce domaine à travers un aperçu de ses parties proches de notre sujet et les outils que nous lui empruntons dans cette section.

Dans les années 90, les ordinateurs dotés d'une puissance de calcul suffisante pour traiter des volumétries d'information importantes se démocratisent. Dans les milieux où existent déjà de grandes bases de données documentaires, dans la banque et l'assurance par exemple, les premiers travaux de fouille de texte voient le jour. Ce domaine « privilégie une démarche *inductive* plutôt que *déductive*, et *numérique* plutôt que *symbolique* » (Tellier, 2011). Cela signifie qu'il s'appuie sur les données qu'il traite plutôt que sur des connaissances a priori, et sur la capture d'information depuis ces données plutôt que sur leur modélisation explicite selon un ensemble de règles. C'est un changement de paradigme très important pour l'extraction de connaissance à partir de documents en général, puisque s'éloignant des systèmes experts jusqu'ici utilisés pour répondre à ces problématiques. Ces nouvelles approches vont se montrer plus à même de capturer l'information implicite contenue dans les jeux de données. Elles distancent ainsi les systèmes experts fonctionnant avec des règles symboliques strictes, limités aux usages couverts par leurs règles.

Il existe plusieurs grandes tâches au sein de la fouille de texte. La RI a pour but de « retrouver un ou plusieurs document(s) pertinent(s) dans un corpus, à l'aide d'une requête plus ou moins informelle » (Tellier, 2011). Pour répondre à ce besoin, un système doit indexer les documents auxquels il a accès, traiter la requête qu'on lui formule et finalement

identifier le ou les documents qui correspondent à cette requête. Cette tâche particulière est plus ancienne que les applications de la fouille de texte au web, en étant par exemple le support des logiciels documentaires servant à indexer les documents numérisés d'archives et de bibliothèques. L'indexation est le procédé permettant d'organiser un référencement de documents en fonction de leur contenu dans le but de les retrouver facilement en fonction de celui-ci.

L'Extraction d'Information (EI) regroupe des méthodes dont l'objectif est « the extraction of specific information from text documents » (Hotho et al., 2005). Le cadre de cette tâche est plus contraint au sens où les informations extraites servent à « remplir les champs d'un formulaire pré-défini » (Tellier, 2011). Une de ses principales applications est la reconnaissance d'entités nommés, qui consiste en l'identification des mots désignant des entités du monde réel (personnes, lieux, organisations, etc) dans un document.

La dernière grande tâche est la classification de texte, qui est « a mining method that classifies each text to a certain category » (Irfan et al., 2015). Les règles permettant cette catégorisation supervisée sont soient fournies au système soient apprises à partir des données. La catégorisation peut aussi être non-supervisée en groupant les documents selon leur similarité et on parle alors de clustering. La catégorisation est une tâche hautement diversifiée, qui trouve des applications dans beaucoup d'autres domaines que la fouille de texte. Dans celui-ci, la détection de spam et le groupement de documents selon les mots qu'ils contiennent sont des exemples d'applications.

1.2.2 RI, EI, Catégorisation, TAL : notre positionnement

À diverses articulations de nos travaux, les enjeux et outils de la fouille de textes sont mis à contribution ou empruntés. Nous ne pouvons pas mobiliser des méthodologies relevant de l'extraction d'information puisque dans le cadre de nos travaux nous ne connaissons pas à l'avance les éléments à extraire. Nous découvrons les tweets qui parlent de l'évènement cible et leur contenu au moment où nous les traitons. La volatilité de la langue sur la plate-forme limite fortement l'efficacité de l'utilisation de ressources linguistiques externes comme des dictionnaires. Comme dit précédemment, nous ne pouvons faire appel à de la classification supervisée puisque nous ne disposons pas de ressource a priori pour traiter les messages que nous captons, qu'il s'agisse d'ontologies ou de données annotées. La catégorisation non supervisée, le clustering, est un élément central de l'approche que nous développons. C'est grâce à cette méthodologie que nous regroupons les messages proches pour identifier les sous-évènements de la cible sur Twitter et nous la présentons de manière approfondie dans le chapitre 2.

La tâche de RI « is the finding of documents which contain answers to questions and not the finding of answers itself » (Hotho et al., 2005). Comme notre approche passe par l'isolation d'ensembles cohérents de messages du reste du corpus, nous nous rapprochons de cet aspect. Toutefois, la question à laquelle nous répondons est très ouverte puisque pouvant être formulée en "qu'est-ce qu'il se dit sur Twitter de l'évènement qui nous intéresse ?" et il n'est pas possible d'y répondre de manière claire en renvoyant un ou quelques tweets. C'est pour cela que nous proposons de décrire des ensembles de tweets cohérents, nous positionnant très poche de la RI sans en faire complètement partie.

L'indexation de document est la méthodologie à la base des moteurs de recherche. Pour retrouver facilement un document en fonction des mots qu'il contient, on peut par

exemple établir l'index inversé d'un corpus qui « indique, pour chaque terme d'indexation, les documents du corpus où il apparaît » (Tellier, 2011) en utilisant les mots du lexique du corpus comme termes d'indexation. L'algorithme du PageRank (Brin and Page, 1998) est un exemple d'indexation de page web basée sur les liens entre pages internet plutôt que sur les mots qu'elles contiennent. Il repose sur le fait de pondérer une page par le nombre de liens permettant d'y accéder depuis d'autres pages, permettant ainsi de retrouver facilement celles qui sont beaucoup référencées.

Nos travaux ont pour but d'identifier des éléments utiles pour décrire des évènements au sein de tweets, sans avoir besoin de savoir de quel tweet en particulier provient un de ces éléments. Les méthodologies d'indexation ne nous sont donc pas utiles. De plus, notre objectif est plus fin que la simple indexation. Nous souhaitons être à même de pouvoir représenter mots et documents avec des coordonnées spatiales qui permettent de les postionner par rapport aux autres mots et documents.

L'extraction d'information ayant pour objectif de sélectionner des éléments bien particuliers au sein de documents textuels, elle s'appuie souvent sur des ontologies. Selon (Stevens et al., 2000), « an ontology should [...] be a conceptualisation of the domain ». Dans cet article, les auteurs expliquent comment est construite une ontologie de bioinformatique, comment en construire une et comment l'exploiter pour extraire des informations d'un texte de ce domaine comme les noms de molécules ou réactions d'intérêts. Comme nous découvrons les évènements que nous souhaitons décrire et comme leurs différents aspects ne sont pas nécessairement connus à l'avance, nous ne souhaitons pas nous appuyer sur des ontologies qui pourraient orienter la collecte des données.

La classification trouve de nombreuses applications en fouille de texte. Dans (Dawei et al., 2021), les auteurs passent en revue des techniques de reconnaissance de sentiment et la classification des documents selon ces derniers. Les méthodes abordées reposent en grande partie sur des modèles d'apprentissage supervisés, présentés en section 1.3.2. Ces derniers ont besoin de jeux de données d'entraînement annotées, où le résultat à trouver pour un document est ainsi connu. Les langues et cas d'usage que l'application de nos travaux concerne ne sont pas dotés de la même manière dans ce type de ressource. Nous faisons donc le choix de ne pas recourir à des méthodes supervisées.

Regrouper des documents selon une mesure de leur similarité est un axe important de la fouille de texte et un aspect critique pour nos travaux. Il s'agit d'un pan de recherche à part entière, que nous avons largement exploité. Nous consacrons donc le chapitre 2 à sa présentation détaillée.

Les présents travaux traitent de corpus délimités automatiquement à partir de la plateforme cible, ici Twitter et son système de requêtes permettant de récupérer un ensemble fini de documents. Ces documents, qui délimitent nos objets d'études, sont formulés en langue naturelle. Pour les raisons que nous avons présentées, c'est une modalité de données particulièrement complexe à traiter. Nos corpus pourraient par exemple être abordés purement par le biais de la statistique et fournir des informations comme l'exemple 1.2.1. Mais ce type d'approche fournit une information trop imprécise, voire ambiguë, et notre objectif est de traiter nos documents de manière plus précise.

Exemple 1.2.1. Dans notre jeu de données tempête Alex comportant environ 20000 tweets, les 10 mots les plus fréquents en dehors des mots-clefs de la requête (voir Partie 1.1.2) sont `alpes-maritimes` (1793 occurences), `après` (1740), `vent` (1424), `aller` (1367), `faire` (1216), `france` (957), `bratagne` (904), `personne` (790), `dégât` (770) et `nuit` (769).

Cette information de fréquence est un outil simple de recherche d'information permettant de se faire une première idée d'un jeu de documents textuels.

Pour pouvoir traiter des données en langue naturelle et fournir une analyse plus fine, la Fouille de Texte fait appel à des outils venant du domaine du TAL (Traitement Automatique du Langage). Ce pan de la recherche est à l'intersection entre la linguistique, l'informatique, les mathématiques et les sciences sociales. Il regroupe plusieurs domaines, allant de la syntaxe aux statistiques, et constitue un point de rencontre entre des travaux de linguistique théorique et des approches ingénieriques plus proches des applications industrielles (Cori and Léon, 2002). Nous y empruntons outils et méthodologies pour circonscrire nos objets d'études.

1.2.3 Représentation de texte : du texte au vecteur

Travailler en s'appuyant sur Twitter implique de traiter d'importants volumes de données. Cette masse pousse à travailler sur une représentation numérique, permettant de passer le traitement des données à l'échelle de leur volume sans avoir à se référer à un lexique par exemple. Notre seul a priori théorique sur la nature de nos données est que les mots sont mis ensemble dans un document à dessein et qu'un document représente ainsi une unité de contexte cohérente et cohésive. Nous avons donc besoin de représentations des mots et de documents qui capturent cette interaction.

Le changement de représentation des données textuelles de leur forme textuelle à une forme numérique est un des problèmes fondamentaux du TAL. L'objectif est d'avoir un format qui permette un traitement à la machine efficace. Cela permet en outre d'exploiter ces données avec des outils et des techniques venant d'autres domaines que le TAL. Ces représentations doivent répondre à plusieurs critères pour être robustes. Comme elles servent de bases pour répondre à une grande diversité de tâches du TAL, leur qualité s'estime la plupart du temps avec les performances que la représentation en question permet d'atteindre pour la tâche donnée. Il est également important pour certaines applications que les représentations soient peu coûteuses en mémoire, et pour d'autres qu'elles permettent des calculs efficaces. La transition entre la représentation numérique proposée et le format textuel des données doit s'opérer sans perdre d'information, ou le moins possible. Une perte peut toutefois être acceptable si elle ne pose pas de problème pour la tâche qui va s'appuyer sur la nouvelle représentation.

Une façon simple de considérer un échantillon de donnée textuelle est de la considérer simplement comme un ensemble de mots (Manning et al., 2008), sans considérer les relations entre ceux-ci. C'est le concept du sac de mots, qui n'observe pas l'ordre dans lequel apparaissent les mots mais prend en considération la présence de chacun au sein des documents. Nous présentons dans cette section cette approche à travers deux exemples largement utilisés qui permettent de comprendre son fonctionnement.

Encodage one-hot Il est possible d'encoder un texte en un vecteur de valeurs binaires, où chaque valeur représente la présence ou l'absence dans ce texte d'un mot du vocabulaire de l'ensemble du jeu de données étudié. On parle d'encodage one-hot (Marcilio, 2019), qui permet d'obtenir une variable à 2^n états codés sur n bits binaires. Par exemple, pour représenter trois valeurs $\{pomme, poire, prune\}$ sur trois bits, un encodage one-hot peut être tel que présenté que le tableau 1.2.

TABLEAU 1.2 – Exemple d'encodage one-hot

valeur	encodage
pomme	100
poire	010
prune	001

Soient D un corpus de documents et L son lexique de taille l, qui est défini par l'ensemble des mots présents dans le corpus dans leur ordre d'apparition. Chaque document $d \in D$ est représenté par un vecteur $(d_1, \ldots, d_l)$, où $d_i = 1$ si le i-ième mot du lexique est dans d et 0 sinon. Cela peut par exemple permettre d'obtenir l'exemple du Tableau 1.3.

TABLEAU 1.3 – Corpus, lexique et représentation par sac de mot. À noter que "pomme" et "pommes" sont considérés comme une même entrée lexicale.

documents	lexique								vecteur
	"je"	"mange"	"une"	"pomme"	"cuis"	"tarte"	"aux"	"John"	
"je mange une pomme"	1	1	1	1	0	0	0	0	$(1,1,1,1,0,0,0,0)$
"je cuis une tarte aux pommes"	1	0	1	1	1	1	1	0	$(1,0,1,1,1,1,1,0)$
"John mange une pomme"	0	1	1	1	0	0	0	1	$(0,1,1,1,0,0,0,1)$

Cette méthode présente l'avantage de permettre d'obtenir rapidement une représentation numérique simple à partir de texte. Toutefois, elle présente de nombreuses limitations. Par exemple, ce changement de représentation ne permet pas de représenter l'ordre des mots dans leur document d'origine, ni le nombre de fois où ils y apparaissent ou leur contexte. Autre limitation majeure, les vecteurs obtenus sont de très grande taille car aussi grand que le lexique, ce qui rend les processus qui les exploitent lourds et coûteux.

Vecteur de scores TF-IDF Pour amoindrir la perte d'informations entre les données d'origine et les vecteurs qui les représentent, il est possible d'utiliser le score TF-IDF (Wu and Salton, 1981) des mots plutôt que des valeurs binaires dans le vecteur représentant un document. Ce score permet de repérer les mots importants dans un jeu de données, avec une importance mesurée en fonction de la fréquence et de la rareté d'un mot. En effet, si un mot est fréquent ou rare dans un corpus, son score est élevé. Réciproquement, s'il est trop fréquent ou trop rare, son score est faible. Ce score permet d'étudier l'importance d'un mot au sein d'un corpus d'un point de vue statistique. C'est un outil venant du domaine de la fouille de texte dont la valeur se calcule comme suit.

Définition 1.2.1. Soit D l'ensemble des documents du corpus étudié avec son lexique L de taille l, le vecteur $\vec{d}$ représentant le document d est :

$$\vec{d} = (s_{d1}, s_{d2}, s_{d3}, ..., s_{dl}) \tag{1.1}$$

Chaque composante de $\vec{d}$ est

$$s_{di} = TF - IDF(i, d, D)$$

avec :

$$TF - IDF(i, d, D) = f_{i,d}.log(\frac{|D|}{|\{d' \in D : i \in d'\}|}) \tag{1.2}$$

$f_{i,d}$ est la fréquence du mot i dans le document d et $\{d' \in D : i \in d'\}$ l'ensemble des documents dans lesquels le mot i apparaît.

Cette formule se compose de deux opérandes : le TF à gauche et l'IDF à droite. La première augmente quand le mot est fréquent dans son document et la seconde quand le mot est rare dans le corpus. À l'échelle d'un document, cela permet par exemple de repérer les mots qui se distinguent de ceux nécessaires pour la construction de la langue comme les "stopwords" (ou mots vides) comme "ce" ou "et", qui apparaissent très souvent peu importe le document. Cette classe regroupe les mots qui n'apportent que peu d'information sur le sens d'un énoncé (i.e. pronoms, déterminants, etc).

En calculant ce score pour chaque mot d'un document, il est alors possible de remplacer les valeurs binaires des vecteurs one-hots pour obtenir des vecteurs de TF-IDF représentant les documents. C'est une amélioration notable des représentations par vecteurs, qui représentent maintenant une partie de l'information à propos de la distribution des mots dans le texte et permettent de mieux profiler les documents.

Tableau 1.4 – Corpus, lexique et représentation par vecteur de score TF-IDF. À noter que "pomme" et "pommes" sont considérés comme une même entrée lexicale.

documents	scores TF × IDF								vecteur
	"je"	"mange"	"une"	"pomme"	"cuis"	"tarte"	"aux"	"John"	
"je mange une pomme"	$1 \times 3/2$	$1 \times 3/2$	$1 \times 3/3$	$1 \times 3/3$	$0 \times 3/1$	$0 \times 3/1$	$0 \times 3/1$	$0 \times 3/1$	$(3/2, 3/2, 1, 1, 0, 0, 0, 0)$
"je cuis une tarte aux pommes"	$1 \times 3/2$	$0 \times 3/2$	$1 \times 3/3$	$1 \times 3/3$	$1 \times 3/1$	$1 \times 3/1$	$1 \times 3/1$	$0 \times 3/1$	$(3/2, 0, 1, 1, 3, 3, 3, 0)$
"John mange une pomme"	$0 \times 3/2$	$1 \times 3/2$	$1 \times 3/3$	$1 \times 3/3$	$0 \times 3/1$	$0 \times 3/1$	$0 \times 3/1$	$3 \times 3/1$	$(0, 3/2, 1, 1, 0, 0, 0, 3)$

Dans le tableau 1.4, on peut voir l'intérêt de la pondération du TF-IDF. Le déterminant "une" a un score moins élevé que des mots plus importants pour séparer les documents les uns des autres, comme "John" et "tarte". On constate également que sur des documents courts comme ceux de cet exemple, le décompte d'un mot dans le document est toujours à 1 ou 0.

Cette représentation présente toutefois certaines des limitations de l'encodage one-hot, ne disant rien par exemple de l'ordre des mots dans le document ou de leur contexte d'apparition, et étant toujours de très grande taille.

Pré-traitements Le texte dans son état brut est difficilement traitable à la machine. Nous avons vu dans la section précédente 1.2.3 comment représenter du texte avec des vecteurs en les considérant comme des ensembles de mots. Les pré-traitements sont une partie cruciale à la base de presque toutes les techniques de traitement de texte à la machine. En plus de permettre un traitement efficace en temps et en mémoire par une machine, les pré-traitements améliorent notablement les résultats des outils qui traitent ensuite ces données préparées en rendant l'information contenue dans le texte plus accessible ou en la complétant. Dans (Irfan et al., 2015), les auteurs proposent de classer les méthodologies de de texte en deux catégories : l'extraction et la sélection de caractéristiques.

Dans ces travaux, Irfan et al. regroupent dans la première catégorie, l'extraction, des méthodologies pour identifier quels éléments dans les textes seraient susceptibles d'en devenir des dimensions de représentation vectorielle. Pour constituer les entrées du lexique

d'un corpus, et donc d'identifier les dimensions de sa représentation par sacs de mots, on peut tronquer les mots pour n'en garder que les racines ou ramener les différentes flexions d'un mot à une forme canonique. On parle de racinisation (ou "stemming") dans le premier cas et de lemmatisation dans l'autre. L'algorithme de racinisation de Porter (Porter, 1980) s'appuie sur un ensemble de règles de dé-suffixation et de suffixes connus pour une langue, pour ramener les mots à ce qu'il considère comme leur racine. Par exemple, le mot "généralisation" serait traité comme suit : `généralisation > généraliser > général > génér`. La lemmatisation de son côté consiste souvent à reposer sur un dictionnaire de formes des mots, comme le *Lefff* (Sagot, 2010) pour le français, pour directement ramener les mots du corpus à une forme considérée comme canon. Par exemple, "créées" serait ramené à "créer". L'intérêt de ces deux techniques pour faciliter la récupération d'information est discuté dans (Balakrishnan and Lloyd-Yemoh, 2014). Les auteurs comparent les résultats d'un moteur de recherche d'une base de documents auquel des requêtes sont envoyées après qu'elles aient été racinisées, avec le PorterStemmer (Hooper and Paice, 2005), ou lemmatisées, avec le lemmatiseur LemmaGen (Jursic et al., 2010). Leurs résultats montrent que les deux techniques améliorent dans le même ordre de grandeur la pertinence par rapport à la requête des documents renvoyés, la lemmatisation permettant d'obtenir des résultats légèrement meilleurs.

Il est également possible d'enrichir la représentation d'un texte avec des informations syntaxiques, grammaticales ou sur les sentiments qu'il véhicule, mais ne mobilisant pas ces mécanismes, nous ne les présentons pas.

1.3 Plongement neuronal de texte

Pour obtenir des représentations plus raffinées des données textuelles, il est possible de faire appel à des outils venant du monde de l'intelligence artificielle : les réseaux de neurones. Une de leur principale application est la construction d'encodages de données et ils sont particulièrement appropriés pour encoder du texte dans des vecteurs. Nous présentons dans cette section les réseaux de neurones et leur application à la construction de représentation vectorielle de texte.

1.3.1 Sémantique distributionnelle

La sémantique est une branche fondamentale de la linguistique qui se concentre sur l'étude de la signification dans le système d'une langue. Elle vise à comprendre comment les mots, les phrases et les énoncés acquièrent et communiquent leur sens dans un système linguistique donné. La sémantique cherche à analyser les significations lexicales et compositionnelles des éléments d'un énoncé, ainsi que les relations qui les lient. Elle propose des outils et des méthodes d'analyse pour examiner les règles et les structures qui régissent la signification linguistique.

Il existe différents axes par lesquels la sémantique peut être abordée (Tellier, 2005). Par exemple la sémantique compositionnelle envisage la construction du sens à partir des significations des mots d'un énoncé et de leur composition avec les règles syntaxiques qui régissent la langue. La sémantique cognitive considère que le sens ne se déduit pas seulement du lexique et de la syntaxe d'un énoncé mais également à travers un ensemble de connaissances, d'expériences et de schémas conceptuels.

La sémantique distributionnelle considère que le sens des mots peut être déterminé en examinant statistiquement leurs co-occurrences, c'est-à-dire en relevant quels mots apparaissent les uns avec les autres. Un « essential distributional fact is that some elements [...] are similar in the sense that if we group these similar elements into sets ("similarity groupings"), the distribution of all members of a set (in respect to other sets) will be the same as far as we can discover » (Harris, 1954). Réciproquement, « the amount of meaning difference corresponding roughly to the amount of difference in their environments » des mots. Ainsi la sémantique distributionnelle mesure la signification linguistique à travers ces relations statistiques, la régularité dans l'utilisation des mots et leur apparition dans des contextes similaires. « You shall know a word by the company it keeps » (Firth et al., 1968). Elle s'appuie sur la notion de paradigme au sens où elle considère que deux mots ont un sens proche si on les trouve dans les mêmes contextes.

C'est cette sémantique distributionnelle que cherchent à capturer les modèles neuronaux.

1.3.2 Modèle à neurones artificiels

Les réseaux de neurones artificiels ont été pensés pour imiter le fonctionnement des neurones naturels dans le cerveau des animaux. Un neurone naturel reçoit, via des conduits nerveux appelés synapses, des stimuli des neurones qui le précèdent sur son chemin synaptique, pouvant consister en signaux électriques ou chimiques. En fonction de la nature, de l'intensité et de la fréquence des signaux reçus, le neurone envoie un signal modulé aux neurones qui le suivent.

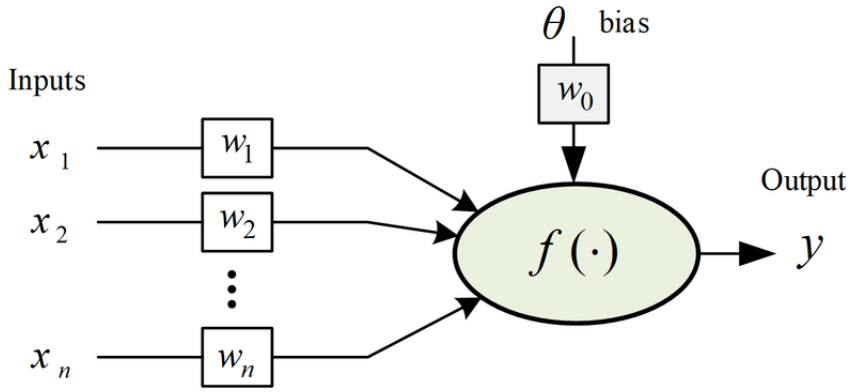


FIGURE 1.1 – Modèle de neurone artificiel, image de (Fathi and Bevrani, 2019). Avec $i \in [1, ..., n]$, étant données les entrées x_i pondérées par leurs poids w_i, un neurone est une cellule de calcul combinant ses inputs $x_i w_i$ avec la fonction f pour calculer un résultat y, en considérant un éventuel biais θ.

C'est ce fonctionnement qui est émulé avec les neurones artificiels. Un neurone artificiel consiste en une unité de calcul, soit une opération, recevant de l'information par ses entrées. Celles-ci ont des poids qui définissent à quel point leur entrée doit être prise en compte pour le calcul à réaliser dans le neurone. La fonction de ce dernier dépend du

comportement qui celui-ci doit reproduire. Dans la figure 1.1 le neurone fait passer ses entrées pondérées $x_i w_i$ dans sa fonction f qui détermine le résultat de ce neurone. Dans la majorité des modèles neuronaux, ce résultat est ensuite envoyé à une fonction d'activation qui détermine s'il doit être produit comme la sortie du neurone, y sur la figure, ou oublié.

Un neurone a pour objectif de produire un bon résultat par rapport aux entrées qu'il reçoit. c'est-à-dire produire un résultat proche de ce qu'on connaît comme étant le "vrai" résultat à produire pour ces entrées. Pour y arriver, on le fait s'exercer sur des données d'entraînement pour lesquelles le résultat à obtenir est connu. Au fil des essais, le neurone met à jour ses poids, les w_i sur la figure 1.1, pour ajuster la considération des entrées et du bruit dans son calcul. Il existe différents types d'apprentissage, le plus répandu étant l'apprentissage supervisé, si un humain a du annoter les données pour définir le résultat attendu, et non supervisé, quand l'objectif du modèle est de découvrir les structures sous-jacentes dans les données. Pour calculer ses mises à jour, le neurone minimise une fonction qui compare sa sortie avec le résultat attendu. Cette fonction est appelée fonction de perte, ou de coût, et peut par exemple être la fonction des moindres écarts carrés :

$$\sum_{i=1}^{n}(y - f(x_i w_i + \theta w_0))^2 \tag{1.3}$$

Les possibilités d'un neurone seul restent limitées. On regroupe donc ces unités de calcul en couches et celles-ci sont empilées pour former des réseaux. Les couches entre celles d'entrée et de sortie sont dites "cachées". Comme chaque couche calcule un résultat à partir de celui transmis par la précédente, on peut les considérer comme des niveaux d'abstraction supplémentaires par rapport aux données d'entrées. Ainsi, plus un réseau compte de couches entre son entrée et sa sortie, plus il peut atteindre un niveau de complexité de résultat élevé. Toutefois, ce gain s'opère au prix d'un entraînement plus long en temps de calcul et nécessitant plus de données pour obtenir un modèle robuste. Il faut donc employer l'architecture et le modèle appropriés pour la tâche que l'on souhaite traiter.

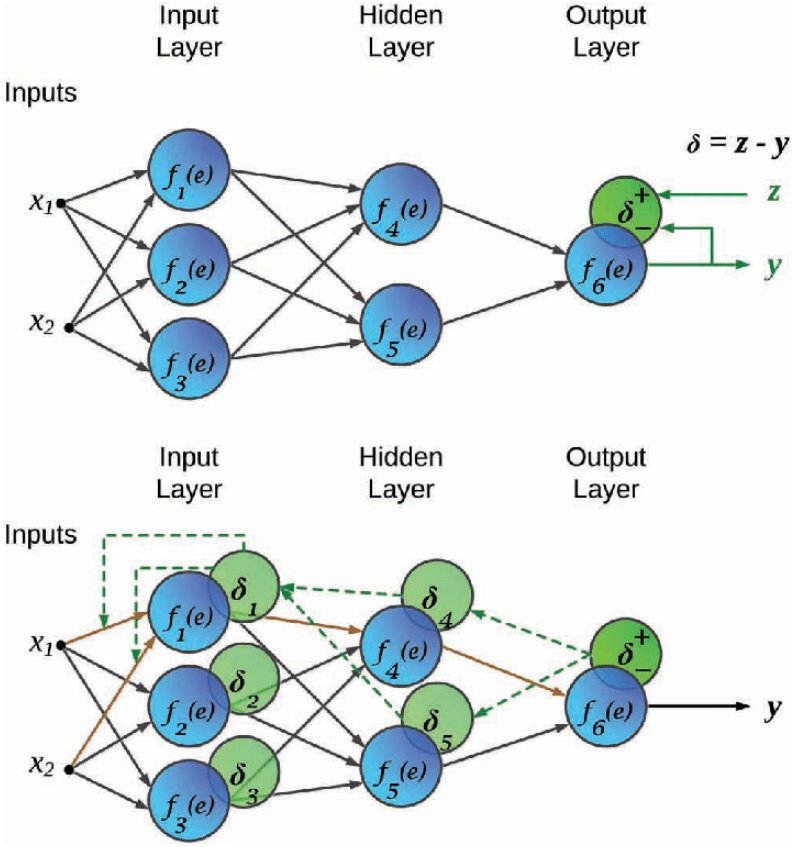


FIGURE 1.2 – Fonctionnement de la rétro-propagation, image de (Onik and Ahmed, 2018). Le modèle calcule une sortie y et la compare au résultat attendu z. Il obtient ainsi son erreur de prédiction à propager δ avec $\delta = z - y$. Les δ_i sur la figure représentent les mises à jour des poids des arcs entre les neurones du modèle.

À l'instar d'un seul neurone, les réseaux ont besoin de s'entraîner pour mettre à jour les poids des connexions entre leurs neurones. La manière la plus répandue de procéder à cet entraînement est au moyen de la rétro-propagation. Cet algorithme proposé dans (Rumelhart et al., 1995) permet de mettre à jour les poids du réseau de manière efficace, comme illustré dans la Figure 1.2. En reprenant la notation de la figure, on peut décrire une étape de l'entraînement du modèle pour expliquer son fonctionnement. Étant donné les inputs x_1 et x_2, le modèle fait une prédiction y. Le modèle compare cette prédiction au résultat attendu z pour calculer la mise à jour δ qu'il va propager dans le réseau avec la soustraction $\delta = z - y$. Les poids des connexions entre les neurones sont mis à jour en fonction de la valeur δ_i que les neurones reçoivent. En répétant plusieurs fois ce fonctionnement sur toutes les instances du jeu de données d'entraînement, le modèle apprend les poids nécessaires pour faire de bonnes prédictions.

1.3.3 Réseaux profonds et auto-encodeurs

Une application notable des modèles neuronaux est la construction de représentation, d'encodage, de données. Soit un modèle avec une couche d'entrée d'une taille x, une ou plusieurs couches intermédiaires de taille plus petite et avec une couche en sortie à nouveau de taille x. En attribuant à ce modèle la tâche de calculer les données d'entrées à partir

d'elles-même avec cette architecture, on lui fait construire au sein de ses couches cachées une représentation des données à partir desquelles il pourra calculer un résultat proche de celles-ci.

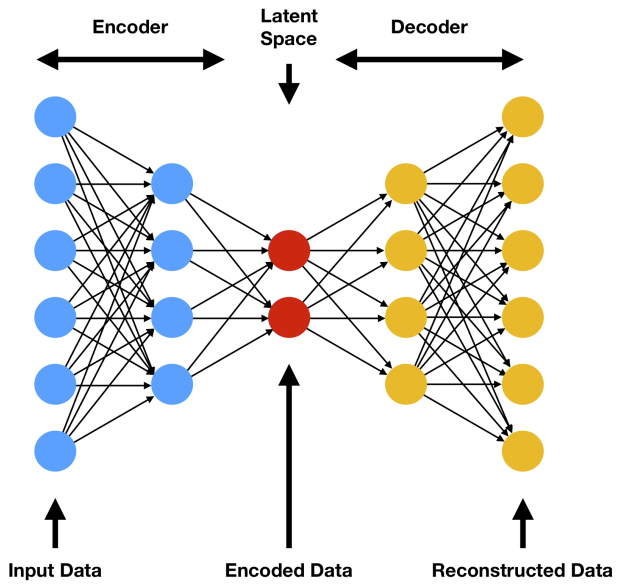


FIGURE 1.3 – Architecture classique d'auto-encodeur (Flores, 2019). Les données d'entraînement fournies en input sont d'abord encodées vers l'espace latent dans la couche cachée centrale du modèle. Puis, le modèle essaie de reconstruire ces données depuis la représentation qu'il a compressée dans cet espace latent.

Le modèle apprend à retenir les parties les plus significatives des données, celles qui lui permettent de reconstruire les instances qu'on lui présente le plus fidèlement possible. Ces informations sont encodées dans l'espace latent du modèle, qui consiste dans sa couche cachée centrale entre ses couches d'encodage et celles de décodage. Cette couche centrale doit être de taille inférieure ou égale à toutes les autres. Plus on diminue la taille de l'espace latent, plus on réduit la dimension de la représentation obtenue. On la récupère en faisant encoder au modèle entraîné toutes les instances d'entrée. Il s'agit ici d'une tâche non-supervisée puisque le modèle compare directement le résultat qu'il produit aux données qu'on lui fournit.

La dimension des représentations est un problème central dans une grande diversité de domaine. Si on prend l'exemple des représentations par sac de mots présenté dans la partie 1.2.3, les vecteurs obtenus sont par exemple aussi grands que le lexique du jeu de données traité. Notons l la taille du lexique d'un corpus de N documents. Un calcul de similarité entre des vecteurs sac de mots coûte l^2, et calculer la similarité entre chaque document d'un corpus représenté ainsi coûte $l^2 \frac{N(N-1)}{2}$. En notant t la taille des vecteurs de représentations appris par un modèle neuronal, ce coût devient $t^2 \frac{N(N-1)}{2}$, où t est en règle générale plus petit que l d'un facteur 10, 100 voire jusqu'à 10 000 dans certaines applications. L'intérêt de la réduction de dimension prend alors tout son sens.

Comme dit précédemment, plus on empile de couches pour former un réseau de neurones, plus grande est la complexité des résultats qu'il peut obtenir. Les modèles les plus massifs peuvent voir leur profondeur se compter en dizaines voire en centaines de niveaux.

1.3.4 Plongement de mots et de documents

Les réseaux de neurones permettent de représenter les données textuelles avec des vecteurs dont la dimension est contrôlée et compacte, qui permettent d'encoder, au moins en partie, la sémantique des mots et des documents représentés. Ces représentations sont appelées plongements lexicaux et plongements de documents, et les modèles qui les calculent sont en général appelés modèles linguistiques neuronaux ou modèles neuronaux de sémantique distributionnelle.

1.3.4.1 Word2Vec

Word2Vec est un modèle de plongement lexical qui apprend des représentations pour les mots en groupant dans son espace latent les mots qui apparaissent ensembles tout en éloignant ceux qui n'apparaissent pas dans les mêmes contextes.

Dans l'article (Mikolov et al., 2013a) proposant ce modèle, deux architectures distinctes sont présentées : le modèle continu skip-gram (à trou) et le CBoW pour "Continuous Bag of Words". Quelle que soit son architecture, le modèle passe en revue les données sur lesquelles il s'entraîne au moyen d'une fenêtre qu'il glisse sur les documents. La différence, intuitivement, est que le modèle devine un contexte à partir d'un mot avec skip-gram, et un mot à partir d'un contexte pour CBoW. La fenêtre glissante est de taille $2a$, a étant la taille du contexte droit et du contexte gauche d'un mot. En considérant un document d de taille n comme une séquence de mots $[w_1, w_2, ..., w_i, ..., w_n]$, le contexte gauche du mot w_i est la sous-séquence de mots $[w_{i-a}, ..., w_{i-1}]$ et réciproquement le contexte droit est la sous-séquence $[w_{i+1}, ..., w_{i+a}]$. Chaque document est donc complété avec a mots `NULL` avant son premier mot et après son dernier pour que chaque mot puisse être considéré dans un contexte complet.

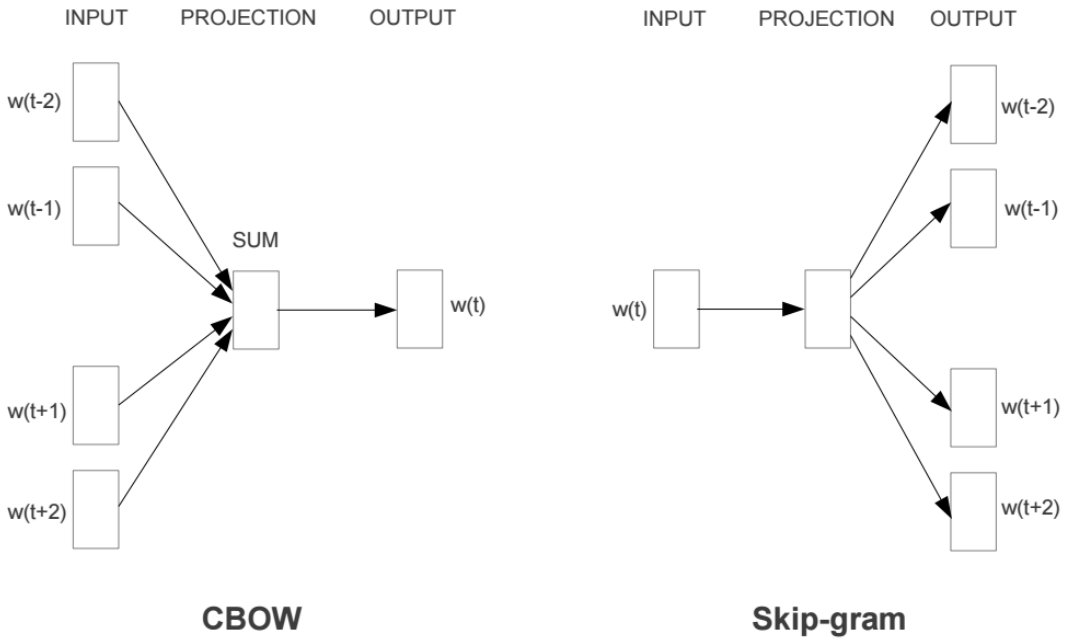


FIGURE 1.4 – Architectures CBoW et Skip-gram du modèle Word2Vec (Mikolov et al., 2013a). Dans son architecture CBoW, le modèle s'appuie sur un contexte pour prédire un mot. Réciproquement, dans son architecture Skip-gram, le modèle s'appuie sur un mot pour prédire un contexte.

Ces architectures s'appuient sur le modèle de calcul de gramme à trou continu ("*continuous skip-gram*"), détaillé dans (Mikolov et al., 2013b). En s'appuyant sur un contexte $c = [w_{i-a}, ..., w_{i-1}, w_{i+1}, ..., w_{i+a}]$ dans lequel apparaît un mot w_i, le modèle cherche à améliorer ses chances de prédire w_i sachant c, soit à maximiser :

$$\sum_{i-a<j<i+a} log\ p(w_i|w_j) \tag{1.4}$$

pour chaque mot du lexique du dataset d'entraînement. Le modèle représente chaque mot w_i avec un vecteur $\vec{w_i}$ et encode les contextes dans lesquels apparaît w_i dans un autre vecteur. Ainsi il peut considérer chaque contexte de la même manière qu'un mot. L'équation 1.4 peut ainsi se réécrire :

$$log\ p(w_i|w_c) \tag{1.5}$$

où w_c représente un contexte dans lequel apparaît w_i et l'objectif du modèle s'écrit :

$$\frac{1}{l}\sum_{i=1}^{l}\sum_{c\in C_{w_i}} log\ p(w_i|w_c) \tag{1.6}$$

où C_{w_i} est l'ensemble des contextes dans lesquels apparaît w_i.

La probabilité est définie avec :

$$p(w_i|w_c) = \frac{\exp \vec{w_i}^\intercal \vec{c}}{\sum_{j=1}^{l} \exp \vec{w_j}^\intercal \vec{c}} \tag{1.7}$$

où l est la taille du lexique L du jeu de données d'entraînement et $\vec{c}$ est le vecteur du contexte passé en revue.

On voit dans ces formules que l'objectif du modèle est de maximiser ses chances de prédire des mots qui apparaissent dans des contextes similaires. Les mises à jour des vecteurs sont calculées par rapport à cette prédiction, permettant de faire des modifications similaires aux vecteurs des mots apparaissant dans des contextes similaires et ainsi de grouper dans l'espace de représentation les vecteurs de ces mots.

La perte du modèle est calculée en comparant le vecteur de probabilité obtenu $p(w_i|w_c)$ au vecteur qui devrait être trouvé, à savoir un sac de mots avec un seul 1 à la position de w_i.

Avec son objectif écrit de la sorte, le modèle continu à trou est coûteux car il a l calculs à faire pour chaque mot du corpus. En écrivant N' le nombre de mots du corpus, cela amène le nombre d'opérations d'une époque d'entraînement à $N'l$. Les auteurs proposent donc de le simplifier avec la fonction logistique et l'échantillonnage négatif (ou NEG pour "negative sampling"). La fonction logistique $\sigma : x \mapsto log(1 + e^{-x})$ permet d'approximer les résultats de la softmax avec un calcul moins coûteux. Le NEG permet de réduire le nombre de prédictions à effectuer avec la méthodologie suivante. Plutôt que de calculer la probabilité d'un mot par rapport à chaque contexte, on se réduit à faire le calcul pour chaque paire mot-contexte apparaissant dans les données. Pour chacun de ces contextes, on tire aléatoirement s autres mots qui n'y apparaissent pas. Ainsi, le nombre d'opérations d'une époque d'entraînement devient $N's$, où on peut négliger s du fait que ce soit une constante paramétrée. Une époque prend donc N' opérations.

L'objectif du modèle peut alors se réécrire en :

$$\sum_{i=1}^{l} \sum_{c \in C_{w_i}} (\sigma(\vec{w_i'}^\intercal \vec{c}) + \sum_{1 \leq j \leq s} \sigma(-\vec{w_j'}^\intercal \vec{c})) \tag{1.8}$$

où C_{w_i} est l'ensemble des contextes dans lesquels le mot w_i apparaît.

Ce modèle de plongement permet de représenter des mots avec des vecteurs compacts, leur dimension étant un des paramètres du modèle à fixer avant de l'entraîner. Sur un des jeux de données que nous avons testé, des représentations en sac de mots comme des vecteurs de scores TF-IDF sont aussi grands que le lexique, étant donc en dimension environ 7 500 (après pré-traitements, voir chapitre 4). Les vecteurs construits avec Word2Vec sont en dimension paramétrée, soit 100, 200 ou 500 en fonction de nos expérimentations. Cela réduit fortement le temps de calcul pour comparer deux mots, par exemple.

Selon la sémantique distributionnelle, les mots que l'on peut commuter en conservant le sens d'une phrase auront tendance à apparaître dans des contextes similaires. Ainsi, on retrouve dans un espace de représentation calculé de la sorte les paradigmes présents dans les données d'entraînement du modèle sous la forme de zones où les mots sont très proches les uns des autres.

1.3.4.2 Doc2Vec

Doc2Vec (Le and Mikolov, 2014) est un modèle de plongement de documents, s'appuyant sur la même architecture que le modèle Word2Vec présenté en partie 1.3.4.1. C'est un modèle peu profond, doté de deux couches, qui apprend en même temps des vecteurs compacts représentant les mots et les documents.

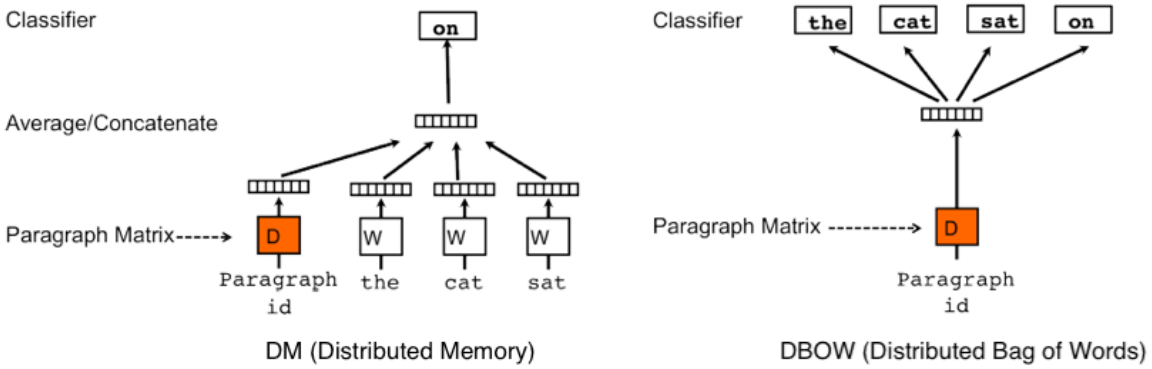


FIGURE 1.5 – Architectures du modèle Doc2Vec proposées dans (Le and Mikolov, 2014)

Sa particularité par rapport à Word2Vec est d'utiliser en tant qu'entrées pour une prédiction non seulement les mots de contexte du mot à prédire, mais aussi le document dont est tiré ce contexte. Les mises à jour des vecteurs sont calculées de la même manière que pour Word2Vec avec le modèle continu à trou avec échantillonage négatif, en intégrant simplement un vecteur supplémentaire au contexte d'un mot.

Les représentations apprises pour les mots sont communes à tous les documents du jeu de données et le modèle apprend un vecteur pour chaque document du corpus. Ainsi, chaque vecteur de document agit comme une mémoire des éléments rencontrés dans ce document avant le contexte qui est en train d'être évalué. En quelque sorte, cette mémoire permet au modèle de "tricher" pour faire ses prédictions par rapport à des contextes, en lui donnant accès à une information en plus.

Avec D un ensemble de N documents, L son vocabulaire de taille l, t la taille fixée des vecteurs à construire et a le nombre de mots constituant un contexte droit ou gauche d'un mot, le modèle apprend trois matrices :

— $\mathbf{D} \in \mathbb{R}^{t \times N}$ pour les documents,
— $\mathbf{W} \in \mathbb{R}^{t \times V}$ pour les mots,
— $\mathbf{G} \in \mathbb{R}^{t(2a+1) \times V}$ pour les contextes.

Son fonctionnement suit l'algorithme 1.

Algorithme 1: Itération d'entraînement de Doc2Vec

input: l'index b du document passé en revue, l'index i du mot à prédire, les vecteurs de mots $\mathbf{W}_j^\intercal$, $\mathbf{W}_k^\intercal$, $\mathbf{W}_l^\intercal$ et $\mathbf{W}_m^\intercal$ du voisinage de L_i dans le contexte tiré de D_b, $\mathbf{D}$, $\mathbf{W}$ et $\mathbf{G}$

1 $J \leftarrow$ une liste contenant l'indice du vecteur de $\mathbf{G}$ associé à L_i et k_{NEG} indices de contextes non associés à L_i

2 $h \leftarrow concat(\mathbf{D}_b^\intercal, \mathbf{W}_j^\intercal, \mathbf{W}_k^\intercal, \mathbf{W}_l^\intercal, \mathbf{W}_m^\intercal)$

3 $e \leftarrow vector(T, 0)$ /* e va contenir l'erreur à appliquer aux vecteurs agrégés dans h */

4 **Pour chaque** $j' \in J$ **faire**

5 $f \leftarrow \sigma(h \cdot G_{j'}^\intercal)$ /* $f \in [0;1]$ est $P(w_j|h)$ avec σ la fonction sigmoid */

6 **Si** $j' = i$ **alors**

7 $l \leftarrow 1$

8 **sinon**

9 $l \leftarrow 0$

/* $j' \neq i$ signifie que la paire $(h, G_{j'}^\intercal)$ est un exemple négatif, soit que le mot associé à la colonne j' de G n'est pas le mot central du contexte à partir duquel h est calculé */

10 $g \leftarrow (l - f) \times \alpha$ /* α = taux d'apprentissage */

11 $e \leftarrow e + g \times G_{j'}^\intercal$

12 $G_{j'}^\intercal \leftarrow G_{j'}^\intercal + g \times h$

13 $\mathbf{D}_b^\intercal \leftarrow \mathbf{D}_b^\intercal + e$

14 $\mathbf{W}_j^\intercal \leftarrow \mathbf{W}_j^\intercal + e$, $\mathbf{W}_k^\intercal \leftarrow \mathbf{W}_k^\intercal + e$, $\mathbf{W}_l^\intercal \leftarrow \mathbf{W}_l^\intercal + e$, $\mathbf{W}_m^\intercal \leftarrow \mathbf{W}_m^\intercal + e$

Dans l'algorithme 1, on peut voir à la ligne 11 que la mise à jour des vecteurs de mots et du vecteur de document est calculée en se basant sur le résultat de la prédiciton de la ligne 5. À la fin de la boucle ligne 4, la mise à jour est composée de l'erreur de prédiction cumulée pour la paire mot-contexte issue des données et pour les k_{NEG} exemples fabriqués. Le signe de la prédiction pour les exemples négatifs est inversé (ligne 10). Ainsi, la mise à jour rapprochera les mots apparaissant ensemble (vraie paire mot-contexte) et éloignera ceux issus d'associations qui n'existent pas dans les données (exemples négatifs). Comme dans la plupart des modèles d'apprentissage neuronaux, la magnitude de ces mises à jour est diminuée au cours de l'entraînement du fait du taux d'apprentissage α qui décroît au fil des itérations. C'est ainsi que le modèle capture, au moins partiellement, la sémantique des données telle qu'elle est présentée dans (Harris, 1954).

1.3.4.3 FastText

Le modèle FastText (Bojanowski et al., 2017) fonctionne sur le même modèle de calcul Skip-gram continu que Word2Vec, avec la même architecture, le même objectif et la même perte. Sa particularité est d'aborder les données textuelles en en exploitant une plus petite unité. Une des limitations de Word2Vec est de ne pas pouvoir inférer de vecteur pour des mots qui ne faisaient pas partie du lexique des données d'apprentissage.

Pour pouvoir s'adapter à un mot jusqu'alors inconnu du modèle, FastText raisonne à base de n-gram de caractères plutôt qu'à base de tokens. Un n-gram est une sous-séquence de n éléments issue d'une séquence donnée. FastText procède ainsi avec les mots en les

considérant comme des n-grammes de caractères.

Exemple 1.3.1. Le modèle FastText paramétré pour fonctionner avec des tri-grammes, calcule une représentation du mot "tempête" en passant par celles des tri-grammes "<te", "tem", "emp", "mpê", "pêt", "ête" et "te>". Les caractères spéciaux '<' et '>' permettent de reconnaître les préfixes et les suffixes.

En partageant les représentations des n-grammes de caractères entre tous les mots, le modèle apprend des briques avec lesquelles il peut s'adapter aux nouveaux mots qu'il rencontre après l'apprentissage. Il en calcule une représentation en s'appuyant sur les n-grammes de caractères qui les composent. Formellement, le modèle calcule les vecteurs de mots comme la somme des vecteurs des n-grammes qui les composent. Si on note $\mathcal{G}_{w_i}$ les n-grammes de caractères d'un mot w_i, alors le vecteur $\vec{w_i}$ est :

$$\vec{w_i} = \sum_{g \in \mathcal{G}_{w_i}} \vec{g}$$

avec $\vec{g}$ le vecteur du n-gram de caractère g.

De la même manière que FastText produit des vecteurs pour les mots en combinant les vecteurs de n-grammes qui les composent, il peut produire des vecteurs de documents en combinant les vecteurs de leurs mots.

1.3.4.4 GloVe

Les plongements présentés jusqu'ici s'appuient sur l'information disponible dans un voisinage de mots pour déterminer leur sens. Dans (Pennington et al., 2014) les auteurs proposent de prendre également en compte des informations statistiques à l'échelle du corpus entier pour calculer des représentations pour les mots.

Le modèle s'appuie sur une matrice globale de co-occurrences Z où chaque cellule Z_{ij} représente le nombre de fois où le mot j apparaît dans le contexte du mot i, le contexte étant de taille paramétrée. Les co-occurrences à l'échelle du corpus sont ainsi prises en compte pour calculer les représentations vectorielles à travers un problème de minimisation des écarts au carré formulé avec :

$$\sum_{J} \sum_{i,j} f(Z_{ij})(\vec{w_i}^{\intercal} \vec{w_j} - \log Z_{ij})^2 \tag{1.9}$$

où J est l'ensemble des contextes dans lesquels les mots w_i et w_j apparaissent ensemble. f est une fonction de pondération de l'information de co-occurrence globale. Selon les auteurs, elle doit satisfaire trois critères : (i) $f(0) = 0$, (ii) f est croissante et (iii) $f(x)$ ne donne pas de trop grands résultats pour les grandes valeurs de x. Ces critères ont pour but de garantir de ne pas sur-pondérer les co-occurrences trop rares ou à l'inverse celles trop fréquentes.

On voit dans cette fonction objectif l'interaction entre les informations globales et locales. La somme sur J garantit de traiter l'information locale en ne limitant les calculs de minimisation des écarts qu'aux contextes où les mots apparaissent ensembles. La pondération par $f(Z_{ij})$ et la soustraction du terme $\log Z_{ij}$ permet de mettre en avant les co-occurrences fréquentes à l'échelle du corpus au moment de calculer les mises à jour des vecteurs $\vec{w_i}$ et $\vec{w_j}$.

Il est possible de calculer des vecteurs de représentations de documents avec GloVe à travers la somme ou la moyenne des vecteurs de mots qui les composent.

1.3.4.5 ELMo

Calculer des représentations de mots communes à tous les documents du corpus d'entraînement pose la limitation de ne pas pouvoir représenter de manière individuelle la pluralité des sens d'un même mot. En reprenant le tableau 1.1, le mot "tempête" serait associé à au moins trois vecteurs avec ELMo, un pour chaque contexte dans lesquels il apparaît dans les trois tweets donnés en exemple. En effet, ce modèle assigne à chaque token de texte une représentation qui est une fonction de la séquence de laquelle il est tiré.

Le modèle consiste en deux LSTM empilés l'un sur l'autre. Un LSTM (acroynome pour "Long Short Term Memory") est un type de réseau de neurones profond dont la particularité est de calculer une sortie par rapport à une entrée donnée et l'état précédent du modèle. Cela lui permet de traiter un élément d'une séquence en se souvenant des éléments qui le précèdent dans celle-ci. Formellement, un LSTM a pour objectif de maximiser la probabilité de prédire un élément d'une séquence sachant les éléments qui le précèdent, soit l'objectif :

$$\sum_{q \in Q} p(q) \tag{1.10}$$

où Q est l'ensemble des séquences du jeu de données traité. $p(q)$ est la probabilité d'obtenir la séquence $q = (t_1, t_2, ..., t_N)$ qui se calcule avec :

$$p(t_1, t_2, ..., t_N) = \prod_{i=1}^{N} p(t_i | t_1, t_2, ..., t_{i-1}) \tag{1.11}$$

Formulé ainsi, on parle de LSTM "forward". La probabilité d'un élément d'une séquence dépend des éléments qui le précèdent. Un LSTM "backward" fonctionne en calculant la probabilité d'un élément en fonction des éléments qui le suivent. L'objectif reste le même en remplaçant le calcul de p avec :

$$p(t_1, t_2, ..., t_N) = \prod_{i=1}^{N} p(t_i | t_{i+1}, t_{i+2}, ..., t_N) \tag{1.12}$$

L'architecture d'ELMo consiste en un LSTM forward et un backward empilés l'un sur l'autre. Le premier produit une représentation des données d'entrées brutes à partir duquel le second calcule la sienne. Finalement, les plongements produits par ELMo sont une combinaison de la représentation brute des données fournies en entrée du modèle et de celles calculées par les LSTM avec la fonction :

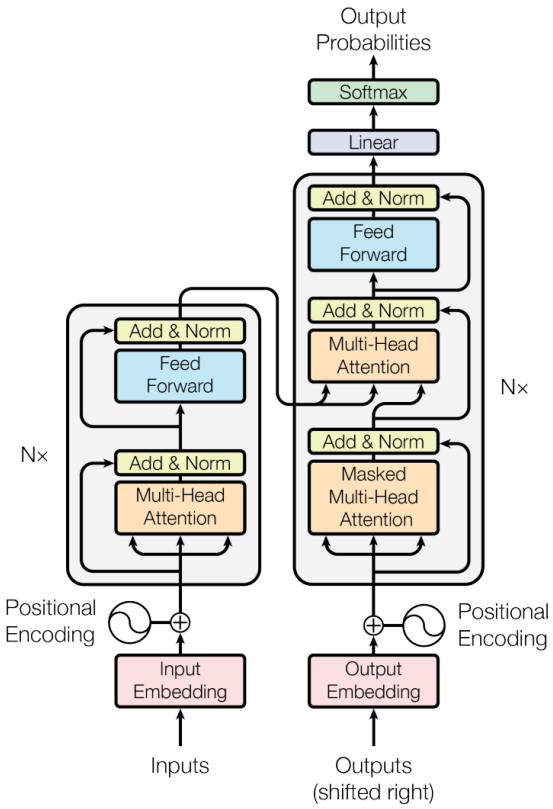
$$ELMo(t) = \gamma \sum_{j=0}^{L} u \cdot h_j \tag{1.13}$$

où γ est un terme de normalisation, u les paramètres de softmax appris par le modèle et h_j la représentation à l'étage j de l'architecture du modèle (h_0 est la représentation des données fournies en entrée de celui-ci).

1.3.4.6 Transformers

Le transformer (Vaswani et al., 2017) est un réseau de neurones profond suivant l'architecture générale des auto-encodeurs. C'est un modèle d'apprentissage supervisé qui est spécialisé sur le traitement de séquence.

FIGURE 1.6 – Architecture du transformer (Vaswani et al., 2017)



Comme on le voit dans la figure 1.6, ce modèle repose sur deux composants : l'encodeur et le décodeur. L'encodeur reçoit les données d'entrées et en calcule une représentation interne au modèle avec un modèle neuronal, représenté avec la brique bleu "feed forward" du schéma. Cette représentation est envoyée au décodeur qui travaille à produire un plongement de sortie et la distribution de probabilité qui y est associée. Le décodeur produit une séquence de sortie par rapport à la séquence d'entrée reçue de l'encodeur en en générant itérativement les éléments. Ainsi, il peut s'appuyer sur les éléments qu'il a déjà générés pour prédire les suivants.

La principale innovation du transformer est d'utiliser l'attention à la place de la récurrence pour traiter les séquences. L'attention est un mécanisme qui permet de pondérer les éléments entre eux au sein d'une séquence (Bahdanau et al., 2014), sans qu'ils soient nécessairement consécutifs. Pour le transformer, elle est définie avec :

$$Attention(Q, K, V) = softmax(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{1.14}$$

où Q, K, V sont des matrices de poids permettant de capturer l'importance des éléments sous la tête d'attention et d_k est la dimension des colonnes de Q et K. La matrice Q

permet d'identifier les parties de la séquence auxquelles accorder plus d'importance et les matrices K et V contiennent les poids permettant de calculer cette importance et sa valeur.

Sur la figure 1.6, on peut voir le mécanisme d'attention mobilisé à trois endroits différents : dans les couches d'encodeurs, entre les couches d'encodeurs et de décodeurs, et dans les couches de décodeurs.

L'attention entre les encodeurs et les décodeurs permettent à ces derniers d'identifier les éléments de l'entrée du modèle sur lesquels se concentrer pour générer les prochains éléments de la séquence. Pour ce faire, un poids est appris pour chaque élément de la séquence issue du décodeur. Ce poids est différent pour chaque position où un même élément serait rencontré. Dans le même temps, cela lui donne accès aux informations de tous les éléments précédents dans la séquence d'entrée. Le mécanisme d'attention interne aux encodeurs et décodeurs est appelé "self-attention" et est une des contributions de (Vaswani et al., 2017). Dans l'encodeur il permet d'avoir accès aux informations, pondérées par leur importance, de tous les éléments de la séquence d'entrée pour en calculer le plongement. Dans le décodeur il permet la même chose, et la partie droite des éléments de la séquence est masquée pour éviter que la génération des nouveaux éléments de la séquence ne s'appuie sur des éléments qui n'existent pas encore. Chaque tête d'attention possède ses propres matrices Q, K et V, et toutes les matrices de toutes les têtes doivent être apprises en même temps que les représentations internes du modèle lors de son entraînement.

1.3.5 Choix du modèle de représentation

Il existe donc dans la littérature une diversité notable de modèles de plongements de documents, et il nous faut choisir celui qui correspondra le mieux à nos besoins.

Nos contraintes sont les suivantes. Puisque nous ne disposons pas de grands volumes de données annotées, le modèle devra être capable d'apprendre des représentations robustes sur des corpus de taille limitée. De plus, il devra pouvoir être entraîné de zéro. En effet, nous ne postulons pas d'a priori sur la langue utilisée dans les données à traiter. Pour être au plus proche de sa réalité dans le discours, le modèle doit donc être entraîné sans connaissance externe, pour nous placer au plus près de cette réalité de terrain. Finalement, la puissance des machines sur lesquelles seront déployés nos travaux est modeste. Le coût computationnel du modèle doit donc rester maîtrisé.

Nous ne nous intéressons donc pas aux modèles dont l'architecture est basée sur celle du transformer. Leur entraînement est coûteux en temps de calcul et nécessite de très grands volumes de données.

Nous avons étudié l'espace appris avec le modèle FastText. Sur nos données, le découpage à une unité de granularité inférieure au mot a tendance à rapprocher les documents en fonction des catégories syntaxiques manifestes qu'ils contiennent. Par exemple, deux documents y sont proches s'ils contiennent des participes présents qui se terminent en "-ant". La topologie sémantique de cet espace n'est pas à la granularité qui nous intéresse.

La méthodologie de ELMo permet d'obtenir des vecteurs de représentations contextuels pour les mots d'un corpus, permettant d'en distinguer les différents sens directement à travers les représentations. Elle peut être étendue à un document, en considérant cette représentation comme une fonction de l'ensemble de ses mots. Il se pose toutefois un problème venant de l'architecture des LSTMs. Il est aujourd'hui connu qu'elle tend à perdre

les informations des séquences qu'elle traite si celles-ci sont trop longues (Wang et al., 2021). Dans le cadre de nos travaux, ce problème est naturellement limité du fait que nous traitions des documents courts. Cependant, les besoins d'entraînement des LSTMs sont trop importants pour nos travaux. Nos corpus sont trop petits et se renouvellent trop vite pour employer ce genre d'approches.

GloVe peut calculer une représentation pour un document à travers la somme ou la moyenne des vecteurs de mots qui les composent. Cela s'appuie toutefois sur l'hypothèse que le sens d'un document peut être réduit à la somme de celui de ses parties. Il s'agit d'une hypothèse forte, dont les fondements théoriques sont discutables notamment parce que le document forme une unité de contexte qui peut changer le sens des mots qu'il contient. De plus, nous avons expérimentalement obtenu des résultats similaires avec les plongements de Doc2Vec et ceux de GloVe, en un temps plus court.

Nous faisons donc le choix d'utiliser Doc2Vec pour apprendre les vecteurs de mots et de documents que nous allons exploiter.

1.4 Évaluation d'une représentation vectorielle de texte

Construire des représentations numériques est nécessaire pour pouvoir traiter des volumes de données textuelles importants. Toutefois, la qualité de ces vecteurs doit être évaluée pour affirmer qu'ils seront utiles pour le traitement de ces données et qu'ils ne le fausseront pas. Il existe deux types d'évaluations des représentations vectorielles de texte et nous en présentons un aperçu dans cette section.

1.4.1 Extrinsèque

La quasi-totalité des modèles de construction de vecteur de représentation de texte n'offre aucune garantie théorique sur la qualité des plongements qu'ils produisent ou sur la nature et la quantité d'information qu'ils encodent. De plus, les vecteurs obtenus sont ininterprétables pour des humains, tant dans la signification de leur dimension que dans les valeurs qui y sont stockées.

Pour contourner cette limitation, on peut utiliser une tâche tierce pour évaluer à quel point un vecteur de représentation de texte permet d'obtenir de bons résultats dans une tâche qui l'exploite. Cette évaluation est dite extrinsèque car elle repose sur un composant autre que les vecteurs eux-mêmes ou que le modèle qui les a produits. Les résultats obtenus sont ensuite évalués avec des métriques classiques telles que l'exactitude *acc* et le rappel *rec*. L'exactitude mesure la proportion d'étiquettes correctes parmi les étiquettes attribuées par le modèle. Le rappel mesure la proportion d'étiquettes correctes attribuées par le modèle par rapport à toutes celles que le modèle devrait attribuer.

$$acc = \frac{TP}{TP + FP} \qquad (1.15) \qquad\qquad rec = \frac{TP}{TP + FN} \qquad (1.16)$$

Dans les équations 1.15 et 1.16, TP correspond au nombre d'étiquettes correctes attribuées par le modèle évalué, FP au nombre de mauvaises étiquettes et FN aux nombre d'étiquettes qui n'ont pas été attribuées.

Différentes tâches ont été proposées en suivant les grandes applications des plongements de texte, avec des données annotées, pour établir un socle commun d'évaluation des modèles. On trouve de nombreux exemples.

(Pang and Lee, 2004) proposent le "polarity dataset" composé de 2000 revues de film annotées avec une étiquette indiquant si le sentiment qui y est exprimé est positif ou négatif, et des ensembles de phrases catégorisées comme subjectives, objectives ou neutres. Cette ressource est par exemple utilisée par (Maas et al., 2011) pour évaluer si le modèle de plongement proposé construit des vecteurs qui permettent de correctement prédire la polarité et la subjectivité d'un texte. (Maas et al., 2011) proposent également un jeu de données annotées basées sur l' "Internet Movie Database" (IMDB), où chaque revue de film est associée à une note par l'utilisateur qui l'a écrite. Pour évaluer si les vecteurs permettent de détecter des compositions de sentiment, (Socher et al., 2013) proposent le "Sentiment Treebank" qui raffine l'annotation proposée par (Pang and Lee, 2004) et l'étend à d'autres revues tirées du même site d'avis cinémas.

Features	PL04	Our Dataset	Subjectivity
Bag of Words (bnc)	85.45	87.80	87.77
Bag of Words (bΔt'c)	85.80	88.23	85.65
LDA	66.70	67.42	66.65
LSA	84.55	83.96	82.82
Our Semantic Only	87.10	87.30	86.65
Our Full	84.65	87.44	86.19
Our Full, Additional Unlabeled	87.05	87.99	87.22
Our Semantic + Bag of Words (bnc)	88.30	88.28	88.58
Our Full + Bag of Words (bnc)	87.85	88.33	88.45
Our Full, Add'l Unlabeled + Bag of Words (bnc)	88.90	88.89	88.13
Bag of Words SVM (Pang and Lee, 2004)	87.15	N/A	90.00
Contextual Valence Shifters (Kennedy and Inkpen, 2006)	86.20	N/A	N/A
tf.Δidf Weighting (Martineau and Finin, 2009)	88.10	N/A	N/A
Appraisal Taxonomy (Whitelaw et al., 2005)	90.20	N/A	N/A

FIGURE 1.7 – Évaluation de l'exactitude d'un classifieur sur trois tâches selon le plongement sur lequel il se base (Maas et al., 2011)

Dans la figure 1.7, on peut voir l'évaluation que ce genre de ressource permet de mener. La colonne la plus à gauche intitulée "Features" indique le plongement sur lequel le modèle se base. Les trois colonnes de droite présentent les résultats en exactitude du modèle de classification sur trois tâches : la classification de sentiment sur la partie analyse de sentiment de la ressource de (Pang and Lee, 2004) dans la colonne "PL04" ; la classification de subjectivité pour les deux autres colonnes, sur la ressource "IMDB" de (Maas et al., 2011) dans la colonne "Our dataset" et sur la partie analyse de subjectivité de la ressource de (Pang and Lee, 2004) dans la colonne "Subjectivity".

Une autre tâche d'évaluation est la prédiction du mot suivant dans un énoncé. (Paperno et al., 2016) propose le jeu de données "LAMBADA" annoté pour évaluer si des modèles linguistiques et les représentations sur lesquelles ils s'appuient permettent de prédire un

mot étant donné une séquence de mot. Le principe de la tâche est relativement simple. Des mots sont cachés au sein de phrases tirées du "Book corpus" de (Zhu et al., 2015). Paperno et al. indiquent que deviner les mots masqués dans leur corpus est difficile même pour des locuteurs natifs de l'anglais, la langue du jeu de données, sans tenir compte des informations venant du contexte précédant la phrase où le mot est caché. Ainsi, cette tâche évalue surtout la capacité d'un plongement à capturer différentes granularités d'information à propos d'un contexte.

La conférence annuelle CoNLL ("Computational Natural Language Learning") s'accompagne d'une tâche partagée à laquelle peuvent répondre les auteurs. Des ressources sont rendues disponibles pour que les auteurs puissent comparer leurs approches pour cette tâche sur la même base. Par exemple, l'édition 2003 (Sang and De Meulder, 2003) proposait un corpus annoté pour permettre d'entraîner, le cas échéant, et de tester des modèles de reconnaissance d'entités nommées.

Ces différentes tâches permettent d'évaluer la qualité des vecteurs produits par un modèle de plongement. Toutefois, des vecteurs peuvent permettre d'obtenir de bons résultats pour une tâche donnée mais pas pour une autre. Par exemple ils peuvent offrir un bon matériau pour détecter des sentiments mais pas pour prédire la suite d'une phrase. Les évaluations extrinsèques sont donc utiles pour savoir si un plongement convient à une tâche, mais ne disent rien des caractéristiques des vecteurs qu'elles permettent d'évaluer.

1.4.2 Intrinsèque

Les évaluations intrinsèques reposent sur la mesure de certains attributs ou de certaines propriétés dans l'espace de représentation. Elles peuvent s'appuyer sur des mesures spatiales, comme la densité, ou sur des comparaisons entre les vecteurs obtenus. Gladkova et Drozd en recensent quatre types dans (Gladkova and Drozd, 2016).

Pour évaluer si le modèle de plongement a correctement capturé la sémantique des mots dans ses données d'entraînement, on peut évaluer la similarité et l'apparentement des mots. Son procédé simple consiste à mesurer la distance entre des mots considérés proches. Par exemple, au plus proches seront les mots "vélo" et "bicyclette" dans l'espace appris par le modèle, au meilleur sera son score pour cette paire. Les jeux de données WordSim-353 (Finkelstein et al., 2002) et MEN (Bruni et al., 2014) permettent de faire des tests pour évaluer cet aspect. La tâche d'analogie de mots, permettant une évaluation similaire, a été proposée dans (Mikolov et al., 2013a) avec un jeu de données permettant de la mettre en oeuvre. Son but est de vérifier que les vecteurs conservent les relations sémantiques et syntaxiques des mots qu'ils représentent à travers la géométrie de l'espace dans lequel ils sont représentés. Par exemple, en notant "$\vec{mot}$" le vecteur de représentation du mot "mot", que l'on ait bien "$\vec{roi}$" − "$\vec{homme}$" + "$\vec{femme}$" = "$\vec{reine}$" et "$\vec{ronronnement}$" − "$\vec{ronronner}$" + "$\vec{gazouiller}$" = "$\vec{gazouillement}$".

Deux autres des types d'évaluations présentés dans (Gladkova and Drozd, 2016) reposent sur une évaluation humaine. Elles sont toutes deux proposées par (Schnabel et al., 2015). La première consiste à entraîner plusieurs modèles différents sur les mêmes données. Ensuite étant donné un mot, on récupère son plus proche voisin dans les espaces appris par les modèles et on demande à des évaluateurs lequel est le plus proche sémantiquement du mot donné. Le meilleur modèle est celui qui donne le plus souvent le meilleur voisin du mot donné. L'autre tâche s'appuyant sur des humains est la détection d'intrus.

Pour un mot donné, on tire ses deux plus proches voisins dans l'espace de représentation appris par un modèle et un autre mot au hasard. Si le modèle a correctement appris la sémantique présente dans ses données d'entraînement, alors un humain doit être capable de détecter l'intrus tiré au hasard parmi les quatre mots d'une itération de cette tâche.

Enfin, une façon d'évaluer la qualité de l'espace appris par un modèle de plongement de texte est d'évaluer si les dimensions qu'il a apprises correspondent à des caractéristiques de la langue de ses données. (Tsvetkov et al., 2015) propose un système permettant de mapper ces caractéristiques, comme des règles syntaxiques ou grammaticales, sur des dimensions des vecteurs de représentations. Le but de cette mesure est de mettre en avant la capture de la structure de la langue par le modèle d'une manière interprétable par un humain.

Les évaluations intrinsèques sont discutables. Par exemple, la tâche d'analogie de mots présentée par (Mikolov et al., 2013a) peut donner un mauvais score pour un modèle appris sur des données qui ne contiennent pas d'exemple permettant de répondre correctement pour les mots des paires du jeu de test. Pour autant, cela ne signifierait pas que le modèle serait incapable de capturer ces relations si ces données d'entraînement les contenaient. Les évaluations reposant sur un regard humain sont toujours sujettes au biais de subjectivité des annotateurs. De son côté, la tâche de mapper des caractéristiques d'une langue sur des dimensions de vecteurs s'appuie sur l'hypothèse que ces dernières ne sont liées qu'à une de ces caractéristiques. Or rien ne dit qu'une dimension n'encode pas de l'information pour plusieurs d'entre elles.

Chapitre 2

Exploration de données

Sommaire

L'exploration, ou fouille, de données est une discipline visant à identifier des caractéristiques d'ensembles au sein de grands volumes de données. Elle regroupe une grande diversité d'outils, s'appuyant aussi bien sur l'intelligence artificielle que les statistiques, et permet de rechercher des structures et des modèles permettant de comprendre et d'exploiter ces larges quantités d'information. Cette façon d'extraire de la connaissance s'est notamment développée avec l'apparition de grands jeux de données provenant des travaux en aide à la décision en entreprise et du développement de la collecte d'informations sur les patients du monde médical, et plus encore par la suite avec les extractions venant du web.

Comme nous l'avons expliqué précédemment, la fouille de texte peut être abordée avec des approches dites supervisées si les caractéristiques des instances à traiter sont connues, ne serait-ce que partiellement, à l'avance. On peut alors utiliser des modèles à base de règles ou qui exploitent des données annotées pour apprendre à répondre à une tâche. Dans le cadre de nos travaux, les données et leurs caractéristiques sont découvertes au moment de traiter les instances. Ainsi, nous ne pouvons pas employer ces méthodologies

et nous nous orientons sur la partie non-supervisée de la fouille de données, aussi appelée clustering.

2.1 Clustering à base de distance

Le clustering, ou partitionnement, est une méthodologie proposant de découvrir des structures sous-jacentes dans des jeux de données. Il permet d'explorer des jeux de données pour lesquels on ne dispose pas ou peu d'informations sur les instances. Il peut s'appuyer sur leurs attributs ou sur des méthodes qui permettent de les comparer.

2.1.1 Définition

Définition 2.1.1 (Partitionnement)**.** Étant donné un jeu de données $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ un algorithme de partitionnement construit, en regroupant les instances similaires, une partition $\mathcal{P} = \{\mathcal{C}_1..., \mathcal{C}_K\}$ composée de K clusters tels que :

— $\forall i \in [1, N], \exists j$ tel que $x_i \in \mathcal{C}_j$

— $\forall i, j \in [1, K], i \neq j : \mathcal{C}_i \cap \mathcal{C}_j = \emptyset$

Pour obtenir la meilleure partition possible, les algorithmes de clustering comprennent un objectif à minimiser qui oriente leur stratégie de regroupement.

Définition 2.1.2 (Clustering)**.** Étant donné un jeu de donnée $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ un algorithme de clustering construit une partition $\mathcal{P} = \{\mathcal{C}_1, ..., \mathcal{C}_K\}$ de K clusters de centroïdes $\{\mu_1, ..., \mu_K\}$ de sorte à minimiser l'objectif :

$$\sum_{i=1}^{K} dist(\mu_i, \{x | x \in \mathcal{C}_i\})$$

Centroïde Le centroïde d'un cluster, noté μ, est un élément important de son analyse. Le barycentre d'un groupement de données permet d'attribuer à celui-ci une position dans son espace de représentation. C'est une moyenne des positions des points d'un cluster. Étant donné un cluster $\mathcal{C}$ regroupant n observations, son centroïde μ est :

$$\mu = \frac{1}{n} \sum_{x \in \mathcal{C}} x$$

La fonction objectif du clustering guide sa stratégie de groupement des instances du corpus à explorer. Elle repose sur la mesure de dissimilarité *dist*, qui peut être adaptée en fonction du problème et de la nature des données traitées.

2.1.2 Mesures de dissimilarité

La manière de calculer la différence entre deux instances d'un corpus diffère souvent d'un corpus à l'autre. Si on représente les instances d'un corpus sur la base des attributs qu'elles ont ou n'ont pas, on peut le faire avec des vecteurs binaires. Ces derniers sont aussi longs qu'il y a d'attributs différents dans le jeu de données et pour une instance x_i, une caractéristique du vecteur vaut 1 si x_i possède cet attribut, 0 sinon. Il est alors

possible de comparer les instances de ce jeu de données avec la distance de Manhattan 2.1.3.

Définition 2.1.3 (Distance de Manhattan)**.** Étant donnés deux vecteurs x_i et x_j dans un espace défini dans $\mathbb{N}^n$, la distance de Manhattan $dMAN$ définie sur [-n ; n] se calcule avec :

$$dMAN(x_i, x_j) = \sum_{f=0}^{n} x_j^f - x_i^f \tag{2.1}$$

Cette distance ne peut toutefois pas se mesurer sur des représentations non-binaires. Lorsqu'on en utilise, il faut alors se tourner vers des mesures exploitant la géométrie de l'espace, comme la dissimilarité du cosinus 2.1.4 ou la distance Euclidienne 2.1.5. Ces métriques s'appuient sur les représentations vectorielles des données dans des espaces de plongements. La première s'appuie sur l'angle formé par les vecteurs des instances, l'autre sur leur écart dans l'espace.

Définition 2.1.4 (Dissimilarité du cosinus)**.** Étant donnés deux vecteurs x_i et x_j dans un espace défini dans $\mathbb{R}^n$, la dissimilarité du cosinus $dCOS$ définie sur [-1 ; 1] se calcule avec :

$$dCOS(x_i, x_j) = 1 - \frac{x_i \cdot x_j}{||x_i||||x_j||} \tag{2.2}$$

Définition 2.1.5 (Distance Euclidienne)**.** Étant donnés deux vecteurs x_i et x_j dans un espace défini dans $\mathbb{R}^n$, la distance euclidienne $dEUC$ définie sur $[0 ; +\infty]$ se calcule avec :

$$dEUC(x_i, x_j) = \sqrt{\sum_{f=0}^{n} (x_i^f - x_j^f)^2} \tag{2.3}$$

La dissimilarité du cosinus est plus appropriée pour comparer des données sans favoriser une caractéristique dont la magnitude est largement plus importante que les autres, là où cet écart de magnitude est directement capturé par la distance euclidienne. Dans un cas de fouille de texte par exemple, où des documents seraient représentés avec des vecteurs formés du décompte des mots qui y apparaissent. Deux documents parlant du même sujet pourraient du fait d'une différence de longueur avoir le décompte d'un de leurs mots notablement différent. Si on s'intéresse seulement à la présence de ce mot, il est préférable de comparer les documents avec la dissimilarité du cosinus. Si on s'intéresse à la représentation statistique des mots, la distance euclidienne est plus appropriée.

Les différentes problématiques de représentations et de mesure de différence entre les instances d'un jeu de données ont engendré l'élaboration de différents algorithmes, mettant en oeuvre différentes stratégies pour en tirer parti. Nous présentons ici différents types de clustering.

2.1.3 K-Means

L'algorithme des K-Means s'appuie sur une représentation spatiale des données qu'il traite pour y détecter des clusters compacts, dont le nombre est paramétré. Après l'initialisation des K centroïdes initiaux, il fonctionne par itérations successives en cherchant à minimiser la distance entre un point et le groupement auquel il est rattaché 2.1.6. Le résultat de K-Means est dit "plat" comme les clusters de la partition obtenue n'ont pas de lien qui permettrait de les organiser, sans hiérarchie ou chevauchement.

Définition 2.1.6 (Objectif K-Means)**.** Étant donné un jeu de donnée $\mathcal{X} = \{x_1, ..., x_K\}$, K-Means construit une partition $\mathcal{P} = \{\mathcal{C}_1, ..., \mathcal{C}_K\}$ de K clusters de centroïdes $\{\mu_1, ..., \mu_K\}$ en minimisant l'objectif :

$$\sum_{i=1}^{K} \sum_{x \in \mathcal{C}_i} dist(\mu_i, x)^2$$

Il s'agit d'un algorithme de clustering paramétré. On lui indique le nombre de clusters qu'il doit construire, il n'a donc pas besoin de découvrir ce nombre à travers son fonctionnement, comme on peut le voir dans l'algorithme 2.

Algorithme 2: K-Means

input : le jeu de données X, le nombre de clusters à construire K
output: le clustering obtenu

```
1 choisir aléatoirement K centroïdes initiaux dans X
2 Tant que non convergence faire
3 |   Pour chaque x ∈ X faire affecter x au centroïde existant le plus proche
4 |   recalculer les centroïdes
```

Le calcul de la convergence s'opère en comparant le résultat de deux itérations successives de l'algorithme, en évaluant la différence entre les clusters obtenus. Il peut être plus ou moins strict en fonction du cas traité, pour être adapté par exemple à la métrique utilisée comme distance au centroïde ou au calcul de ces derniers.

Initialisation et K-Means++ Dans la formulation initiale de K-Means, les premiers centroïdes de clusters sont sélectionnés aléatoirement au sein des données. En plus de rendre l'algorithme non déterministe, cela peut fortement dégrader la qualité de la partition obtenue ou le temps de calcul nécessaire à son calcul si les K premiers centroïdes ne sont pas favorables. Par exemple, s'ils sont tous très proches les uns des autres, la convergence peut devenir difficile et longue à obtenir car un point appartenant à un cluster peut être attribué à un autre cluster lors du recalcul des centroïdes, puis ré-attribué au premier cluster au recalcul suivant. Si les centroïdes initiaux sont tous en bordure de l'espace de travail sauf un plus central, les clusters obtenus seront très déséquilibrés car une majorité d'instances sera assignée au centroïde du "milieu".

Pour limiter cet effet négatif, d'autres méthodes d'initialisation permettant de meilleurs résultats ont été proposées. En particulier la méthode "K-Means++" (Arthur and Vassilvitskii, 2007) est largement utilisée. Elle consiste à sélectionner des centroïdes initiaux

les plus éloignés possibles les uns des autres pour répartir les clusters dans l'espace de représentation des données.

Application à la fouille de texte Le clustering est très largement utilisé en fouille de texte, dans une très grande variété d'applications. Nous nous concentrons sur celles mettant à profit des représentations vectorielles de documents pour identifier des groupes de similarité au sein de corpus. Même en restreignant ainsi le faisceau d'étude, on trouve des travaux de la médecine à la revue de presse. Dans (Curiskis et al., 2020), les auteurs passent en revue des techniques de représentations et de clustering pour identifier les approches les plus adaptées au traitement de données issues du web social. Dans (Saad et al., 2006), Saad et al. comparent des techniques de clustering pour identifier les plus adéquates, et leurs paramètres, pour extraire de la connaissance depuis des documents médicaux.

La capacité du clustering à regrouper des éléments selon une mesure de similarité est aussi simple à appréhender qu'elle est puissante. C'est pour cela que nous la sélectionnons dans le cadre de nos travaux où nous souhaitons regrouper des documents de sens similaire. Dans le reste de cette section, nous présentons d'autres aspects et approches du clustering proches de nos travaux.

2.1.4 Clustering hiérarchique

Il est possible d'adapter le fonctionnement même de la stratégie d'agglomération. K-Means construit par exemple ses clusters en affectant des instances au plus proche centroïde de cluster existant. À la place, on peut considérer initialement chaque instance du corpus comme un cluster et fusionner successivement ces clusters pour arriver à former le nombre de clusters souhaité. On parle parfois d'approches agglomératives, comme elles consistent à agglomérer des clusters. Ce genre d'approche de bas en haut est un peu plus coûteuse en temps mais permet d'obtenir un fonctionnement déterministe, qui procède avec un peu plus de recul sur les données qu'un K-Means.

Algorithme 3: Clustering hiérarchique

input : le jeu de données X, le nombre de clusters à construire K
output: le clustering obtenu $\mathcal{P}$

1 initialiser $\mathcal{P}$ en faisant de chaque point de X un cluster de un élément
2 **while** $|\mathcal{P}| > K$ **do**
3 choisir deux clusters A et B dans $\mathcal{P}$ selon la stratégie de linkage
4 fusionner A et B
5 retourner $\mathcal{P}$

En plus de la mesure de distance entre les instances, le clustering hiérarchique comporte un autre élément supplémentaire par rapport au clustering classique : sa stratégie de linkage. Elle permet de comparer les clusters et de sélectionner ceux à fusionner, comme cela apparaît à la ligne 3 de l'algorithme 3. Il en existe plusieurs, permettant de répondre à autant de problématiques d'agglomération différentes.

Minimum / single linkage Avec cette stratégie, l'algorithme cherche les deux points venant de deux clusters différents qui ont la plus courte distante entre eux et fusionne ces

deux clusters, soit la paire de clusters de :

$$argmin_{\mathcal{P}}\ dist(a \in A,\ b \in B)$$

Cette stratégie est par exemple adaptée dans les cas où les clusters à identifier ne sont pas sphériques.

Average linkage Cette stratégie cherche et fusionne les clusters qui ont la plus petite distance moyenne entre leurs points, soit les clusters de :

$$argmin_{\mathcal{P}} \sum_{a \in A} \sum_{b \in B} dist(a, b)$$

Maximum / complete linkage En fonctionnant presque comme la réciproque du Single Linkage, cette stratégie cherche la plus petite distance entre les points les plus éloignés de deux clusters et fusionne ces derniers, soit la paire de clusters de :

$$argmax_{\mathcal{P}}\ dist(a \in A,\ b \in B)$$

Average et *complete linkage* seront à même de détecter des clusters sphériques, plus compacts avec la première stratégie qu'avec la seconde.

Ward linkage Cette dernière stratégie de clustering hiérarchique fonctionne en sélectionnant la paire de clusters dont la fusion augmentera le moins possible la somme des distances au carré des points au centroïde de leur cluster dans la partition obtenue, soit la paire de clusters de :

$$argmin_{\mathcal{P}} \sum_{x \in A \cup B} dist(\mu_{A \cup B}, x)^2 - \sum_{x \in A} dist(\mu_A, x)^2 - \sum_{x \in B} dist(\mu_B, x)^2$$

Utiliser le *Ward linkage* rend les clusters obtenus similaires à ceux d'un K-Means tout en obtenant une structure arborescente du partitionnement des données étudiées.

Il faut noter que l'algorithme 3 présente une approche de bas en haut ou *agglomérative*. Initialement, chaque point est considéré comme un cluster soit comme une feuille de l'arbre de partitionnement et on obtient le nombre de clusters voulu en fusionnant successivement ces feuilles. La réciproque de ce type de fonctionnement, dite de haut en bas ou *divisive*, fonctionne en partant d'un cluster regroupant toutes les données, de la racine de l'arbre, et en le divisant successivement jusqu'à obtenir K feuilles.

2.1.5 Clustering par densité

Pour mettre à profit la représentation dans un espace de plongement sans se baser uniquement sur la distance que l'on peut y mesurer entre deux points, il est possible de raisonner sur la base de la densité de peuplement de cet espace. L'algorithme DBSCAN (Ester et al., 1996) fonctionne avec deux paramètres : un pour le nombre minimum d'instances au sein d'un cluster et un pour la distance maximum entre deux points pour les considérer voisins. L'idée est de détecter les clusters en suivant les zones de fortes densités

Algorithme 4: DBSCAN

input : le jeu de données X, le nombre minimum de points dans un cluster α, la distance maximum entre deux points voisins ϵ
output: le clustering obtenu $\mathcal{C}$

marquer tous les points de X non vus
initialiser $\mathcal{C}$ à vide
Pour chaque x *non vu* $\in X$ **faire**
    marquer x comme vu
    $v \leftarrow \{x' \in X \ : \ dist(x, x') \leq \epsilon\}$
    Si $|v| \geq \alpha$ **alors**
        $c \leftarrow$ le cluster initialisé avec x
        $Agglomerer(X, \alpha, \epsilon, \mathcal{C}, c, v)$
    sinon
        marquer x comme bruit
return $\mathcal{C}$

Algorithme 5: Agglomerer

input : le jeu de données X, le nombre minimum de points dans un cluster α, la distance maximum entre deux points voisins ϵ, le clustering en cours de construction $\mathcal{C}$, le cluster en cours d'agrandissement c, un ensemble d'instances v

Pour chaque v' *non vu* $\in v$ **faire**
    marquer v' comme vu et l'affecter à c
    $w \leftarrow \{w' \in X \ : \ dist(w', w) \leq \epsilon\}$
    Si $|w| \geq \alpha$ **alors**
        $Agglomerer(X, \alpha, \epsilon, \mathcal{C}, c, w)$
    sinon
        marquer v' comme bruit

comme dans l'algorithme 4, de sorte à détecter le nombre de clusters existants dans les données.

Ce type de stratégie de regroupement permet de détecter des clusters de formes non sphériques, à la différence du K-Means et de certaines stratégies de fusion du clustering hiérarchique. Il nécessite toutefois d'avoir une connaissance fine du cas d'application et des données traitées puisqu'il faut pouvoir définir a priori le seuil à partir duquel on considère deux points trop dissimilaires pour être voisins, ainsi que la taille minimum d'un cluster.

Ce genre de raisonnement peut être abordé avec des graphes, comme par exemple dans la variante OPTICS de DBSCAN. En transformant le paramètre ϵ d'une valeur à un domaine, OPTICS commence par construire un graphe où les noeuds sont les instances et où les arcs relient les noeuds si la distance entre les deux instances concernées se trouve dans le domaine ϵ.

2.2 Clustering conceptuel

Construire des clusters est un objectif en soit. Toutefois si la finalité est de rendre leur contenu compréhensible pour un humain, il faut également être en mesure de produire quelque chose qui permette à ce dernier de se représenter l'objet de ce cluster. Il existe différentes approches pour répondre à ce besoin et on trouve parmi elles le clustering conceptuel, une variation du clustering qui intègre une tâche d'identification et/ou de construction d'un concept permettant de se faire une idée de ce que contient un cluster.

2.2.1 Approches à base d'optimisation

Le clustering conceptuel fonctionne sur des ensembles de données transactionnelles. Dans ce type de jeu de données, une instance est une transaction consistant en une séquence d'items.

Tableau 2.1 – Exemple d'ensemble de données transactionnelles

transactions	items				
t_1			C		E
t_2	A			D	E
t_3		B	C		
t_4	A		C		E
t_5	A	B		D	
t_6	A	B	C		
t_7	A		C	D	

Dans l'exemple donné en figure 2.1, on voit que la transaction t_1 regroupe les items A, B et E. L'objectif du clustering conceptuel est d'identifier des concepts sous la forme de séquences d'items, à partir de leurs motifs d'apparitions émergeant dans les transactions.

Définition 2.2.1 (Clustering conceptuel)**.** Soient un ensemble $\mathcal{I} = \{it_1, ..., it_I\}$ et un ensemble de transactions $\mathcal{T} = \{t_1, ..., t_T\}$, un algorithme de clustering conceptuel identifie K concepts $C_1, ..., C_K$ permettant de construire une partition $\mathcal{P}$ de K clusters disjoints

$\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K$, chaque élément de $\mathcal{T}$ étant attribué à un cluster. Le concept C_i doit permettre d'expliquer la présence des éléments au sein de $\mathcal{C}_i$. L'algorithme optimise une fonction d'évaluation f qui exprime la qualité du clustering.

Selon les spécificités du problème traité, il y a différentes façons de considérer qu'un concept C_i couvre une transaction t du cluster $\mathcal{C}_i$. On parle de couverture pour exprimer le fait qu'un concept explique la présence d'une transaction dans son cluster. Par exemple on peut considérer que tous les items de t doivent apparaître dans C_i pour que ce concept la couvre, ou qu'il en suffit d'un. Cela influe notamment sur la taille des concepts identifiés et sur la complexité du calcul permettant de les obtenir.

Changer l'expression de f permet d'obtenir différents résultats. Dans l'exemple de la figure 2.1, en notant $cov(t, C)$ le fait que la transactions t soit couverte par le concept C avec $cov(t, C) \Leftrightarrow \exists it \in \mathcal{I} | it \in t \wedge it \in C$. Si

$$f = max(|\{t|cov(t, C)\}|) \wedge min_{i \in [1,K]}|C_i|$$

soit si l'objectif est de maximiser le nombre de transactions couvertes par des concepts de taille minimale, alors une solution pour $k = 3$ serait $\{\{B\}, \{E\}, \{A, C, D\}\}$ et les clusters associés $\{\{t_3, t_5, t_6\}, \{t_1, t_2, t_4\}, \{t_7\}\}$. Si

$$f = max(|\{t|cov(t, C)\}|) \wedge max_{i \in [1,K]}|C_i|$$

soit si l'objectif est de maximiser le nombre de transactions couvertes par des concepts de taille maximale, alors une solution pour $k = 3$ serait $\{\{A, D\}, \{B, C\}, \{C, E\}\}$ et les clusters associés $\{\{t_2, t_5, t_7\}, \{t_3, t_6\}, \{t_1\}\}$.

Le clustering conceptuel est formalisé dans (Michalski, 1980). Dans ses travaux, Michalski adresse le problème de compréhensibilité de partitions, ces dernières étant composées de cluster trop volumineux pour qu'un humain puisse les appréhender. À la place d'utiliser des méthodes à base de similarité d'instances qui ne sont pas adaptées à tous les problèmes, il propose de sélectionner des variables pour décrire les instances d'un jeu de données et de les exploiter pour décrire les structures détectées par le clustering.

Michalski propose un cadre théorique pour tirer parti de la connaissance rendue accessible de la sorte et l'algorithme CLUSTER/PAF pour l'implanter. Sa méthodologie s'appuie sur des *complexes*, des ensembles de produits logiques de la forme $[a_j \# v]$ où $\#$ est un opérateur dans $[\neq, \leq, <, >, \geq, =]$ et v une valeur dans le domaine de a_j. Ces complexes sont itérativement modifiés pour que les instances satisfaisant les *complexes* forment des ensembles disjoints en optimisant un assemblage de fonctions d'objectifs et de coûts. Cette approche a été améliorée dans les années qui ont suivi (Michalski and Stepp, 1983; STEPP, 1986; Seeman and Michalski, 2006) mais est fondamentalement trop coûteuse en mémoire et en temps de calcul pour pouvoir être appliquée dans des contextes d'exploration de données textuelles, où le nombre d'attributs et de transactions dans les jeux de données explose.

2.3 Clustering incrémental

Dans la vaste diversité du clustering, il faut distinguer quel point d'une approche comporte un caractère incrémental. Par exemple les approches dites *anytime* fonctionnent

sur de longues durées, pour des raisons de complexité de calcul ou de contrainte forte sur la qualité de leur résultat mais il est possible de mesurer leur résultat à n'importe quel moment. Dans (Lin et al., 2004), un ensemble de signaux sous forme de séries temporelles est clusterisé au cours d'un processus *anytime*, à des degrés de décomposition de signal de plus en plus fin. On peut donc considérer cette approche comme incrémentale car son résultat est construit de manière progressive.

Pour les présents travaux de thèse, nous nous intéressons aux approches qui sont incrémentales par rapport à leurs données, c'est à dire celles « [where] all the data to be clustered may not be available at one time. For instance, applications where data is constantly collected and analyzed to detect differences » (Rowland and Vesonder, 1987). Nous considérons les approches pour lesquelles « [the] output must be available after processing each example ; it cannot wait for all the input » (Lebowitz, 1987).

2.3.1 Fonctionnement à base de distance

Si les données ne sont pas disponibles sous la forme d'un ensemble figé et fini d'instances, par exemple si on les reçoit sous forme d'un flux constant, il n'est pas possible de les explorer avec des algorithmes de clustering classiques. Par exemple, l'ajout continu d'instances empêcherait un K-Means classique de converger puisqu'il devrait a minima recalculer tous ses centroïdes à chaque ajout de données. Les différentes stratégies de linkage d'une approche hiérarchique ont besoin de pouvoir comparer toutes les instances à grouper au moment de la décision de fusion ou de division de cluster. L'ajout à la volée d'une nouvelle instance peut rendre une précédente décision de linkage classique caduque.

Par ailleurs on ne peut définir à l'avance le nombre de clusters dans un jeu de données qui est inconnu par nature puisqu'en cours de constitution. Il faut donc adapter les méthodes à la modalité des flux de données.

Définition 2.3.1 (Clustering incrémental)**.** Étant donnée une source de données S qui fournit en continu de nouvelles instances de données, un algorithme de clustering incrémental a pour objectif de constituer une partition de clusters disjoints $\mathcal{P} = \{\mathcal{C}_1, ..., \mathcal{C}_{K'}\}$ couvrant toutes les instances reçues, avec K' le nombre de clusters qui évolue au fil de son exécution. Ce type d'approche comprend une fonction d'objectif f qui guide l'agrandissement des clusters connus et décide de la création de nouveaux clusters. Leurs applications pratiques doivent également avoir une stratégie de gestion de la mémoire pour éviter d'en déborder et de bloquer leur exécution.

Comme expliqué, toutes les instances ne sont pas connues à l'avance, ni le nombre de clusters à construire. Ainsi, il soit faut pouvoir garantir la possibilité de comparer les nouvelles instances à celles que l'on a déjà, soit que leurs modalités sont invariantes ou contraintes par un ou des filtres. Si ce n'est pas le cas, la mesure de dissimilarité utilisée pour comparer les instances et décider de la construction des clusters doit être robuste face à cette variabilité. En outre, la stratégie décidant de l'agrandissement des clusters connus et au contraire d'en créer un nouveau doit permettre d'optimiser la fonction f.

D'autres problématiques apparaissent avec les flux de données. Par exemple, si on note la mesure de dissimilarité d'un clustering incrémental *dist* (voir section 2.1.2) et que f est similaire à l'objectif d'un K-Means avec :

$$f = min_{\mathcal{C} \in \mathcal{P}} \sum_{x \in \mathcal{C}} dist(\mu_{\mathcal{C}}, x)$$

L'algorithme ainsi formulé optera pour la solution triviale consistant à faire de chaque nouvelle instance un cluster de un élément. Pour éviter un tel résultat il faudrait donc également contraindre la complexité maximum de la partition obtenue, ce qui reviendrait à paramétrer le nombre de clusters à découvrir alors que c'est impossible dans ce genre de problème. Il est de plus prouvé que ce type de limitation dégrade fortement les résultats de clustering incrémental (Ackerman and Dasgupta, 2014).

Il faut donc éviter des formulations aussi directes. La fonction f qu'un tel algorithme embarque formule plutôt le raisonnement permettant de raccrocher une nouvelle instance à un cluster connu. En notant *dist* la mesure de dissimilarité, f sera de la forme :

$$f(x, \mathcal{P}) = \begin{cases} \text{si } \exists\ \mathcal{C} \in \mathcal{P} \mid dist(x, \mu_{\mathcal{C}}) < \alpha & \text{attribuer } x \text{ à } \mathcal{C} \mid min(dist(x, \mu_{\mathcal{C}})) \\ \text{sinon} & \mathcal{P} = \mathcal{P} \cup \{x\} \end{cases} \tag{2.4}$$

On constate alors que ce type de formulation va de paire avec des problématiques de seuil, représentées dans la formule par le paramètre α, qui déterminent la décision de la création d'un nouveau cluster. C'est le pendant incrémental du paramétrage a priori du nombre de clusters des approches classiques. Notons que formulé ainsi, l'objectif du clustering ne modifie pas les clusters déjà existants. Les messages étant reçus de manière temporelle, cela peut entraîner la séparation d'instances qui devraient être dans un même cluster. On parle alors de fragmentation.

Fragmentation Lorsqu'on intègre une nouvelle donnée à une partition existante, on ne peut le faire qu'en considérant les données que l'on connaît déjà. Ainsi, des instances peuvent être raccrochées à un cluster étant le meilleur choix au moment de leur réception. Au fil du temps, des données peuvent ainsi être agglomérées dans des clusters séparés alors qu'ils devraient n'en former qu'un.

Dans la littérature, des approches comme celles de (Becker et al., 2011) ou (Hasan et al., 2019) traitent le problème de façon similaire. Quand les données sont reçues, elles sont comparées à tous les membres de la partition existants à ce moment là. Si une instance pourrait être raccrochée à plusieurs d'entre eux, on garde ces derniers en mémoire comme des candidats à la fusion de cluster. Dans (Hasan et al., 2019), quand le nombre d'instances en commun dans deux candidats à la fusion dépasse un seuil paramétré, ils sont fusionnés. Dans (Becker et al., 2011), la fusion a lieu si plus de la moitié des instances de deux candidats contiennent des éléments identiques.

Utiliser la force brute Une façon simple d'appliquer directement des méthodologies de clustering classique sur des données arrivant en flux serait d'accumuler les instances entrantes et de recalculer la partition complète à intervalles réguliers. Dans (Can, 1993), l'auteur propose de maintenir à jour une matrice C représentant le jeu de documents textuels étudié. Chaque cellule c_{ij} de cette matrice contient la probabilité de trouver un mot du document i dans le document j et est appelée *cover-coefficient*. Cette matrice est mise à jour régulièrement selon un paramètre donné, en y ajoutant de nouveaux mots et en retirant ceux devenus obsolètes. Elle permet d'identifier des centres de clusters,

en permettant de détecter les documents maximisant les chances d'y trouver des termes d'autres documents, et de construire les clusters autour d'eux. Elle doit toutefois être entièrement recalculée à chacune de ses mises à jour.

Ce type d'approche n'est envisageable que dans les cas où le temps de réponse peut être long ou en disposant d'une puissance de calcul et d'une quantité de mémoire très importantes. Dès lors que les données doivent être accumulées en grandes quantités ou si le problème nécessite une solution rapide, il faut changer de paradigme.

2.3.2 Clustering incrémental conceptuel

S'il est possible de transposer la mécanique de clustering plat à base de distance sur des flux de données comme nous venons de le voir, il est également possible de le faire avec celle du clustering conceptuel. Les processus incrémentaux peuvent aussi bien être basés sur une mesure de distance que sur la construction continue de concepts permettant d'identifier les clusters. Dans ce genre de cas, f fonctionne sur les attributs de la nouvelle transaction t qui lui est présentée par rapport à l'ensemble des concepts C déjà connus. Si on note C_i le concept permettant de construire le cluster $\mathcal{C}_i$ et $covC(t, C_i)$ une fonction attribuant un score à la couverture de la transaction t par le concept C_i, on peut écrire f comme :

$$f(t, C, \mathcal{P}) = \begin{cases} \text{si } \exists\ C_i \in C \mid covC(t, C_i) > \alpha & \text{attribuer } t \text{ à } \mathcal{C}_i \mid max(covC(x, C_i)) \\ \text{sinon} & \mathcal{P} = \mathcal{P} \cup \{t\} \text{ et } C = C \cup \{t\} \end{cases} \tag{2.5}$$

S'appuyer sur une méthodologie à base de concept pour faire du clustering incrémental consiste en général à établir les concepts existants au sein d'un jeu de données de départ. Un premier clustering en est déduit et il est possible de recevoir de nouvelles instances. Si elles correspondent de manière satisfaisante à un des concepts connus, elles sont naturellement clusterisées. Dans le cas contraire, elles sont fouillées pour détecter de nouveaux concepts ou augmenter le score de concepts déjà connus mais trop peu représentés pour faire partie du clustering, et faire évoluer la structure du clustering le cas échéant. C'est par exemple le cas dans (Hennig and Wurst, 2006), dont nous parlons plus en détail ci-après.

2.3.3 Application à la fouille de texte

Les flux de données en ligne sont un terrain d'application tout trouvé pour le clustering incrémental, s'inscrivant dans diverses problématiques scientifiques allant de la veille de presse (Petrovic et al., 2013) à la réponse de crise face à des catastrophes naturelles (Yin et al., 2012). La capacité à ingérer de nouvelles données est un élément central de l'écoute de flux RSS ou Twitter, par exemple.

2.3.3.1 Clustering incrémental plat

Utiliser des approches à base de distance permet de construire des partitions avec les instances reçues en mesurant directement leur éloignement aux centroïdes connus.

Dans (Becker et al., 2011), les auteurs traitent du problème de détection d'évènements basée sur Twitter. Leur objectif est de différencier les messages à propos d'évènements et

ceux qui ne le sont pas. Pour ce faire, Becker et al. proposent d'employer un algorithme incrémental, décrit dans l'algorithme 6.

Algorithme 6: Construction incrémentale de cluster de (Becker et al., 2011)

```
   input: le flux Twitter F, un seuil de similarité μ
1  P ← ∅ Pour chaque nouveau tweet t ∼ F faire
2      t⃗ ← TF − IDF(t)
3      C′ ← NULL /* C′ est le cluster auquel on va assigner t        */
4      Pour chaque cluster C ∈ P faire
5          Si dCOS(t⃗, C⃗) ≤ μ alors
6              Si C′ = NULL alors
7                  C′ ← C
8              sinon
9                  Si dCOS(t⃗, C⃗) ≤ dCOS(t⃗, C⃗′) alors C′ ← C;
10     Si C′ = NULL alors
11         initialiser un nouveau cluster contenant t
12     sinon
13         affecter t à C′
```

Chaque tweet reçu du flux de données est représenté avec un vecteur composé des scores TF-IDF des mots qu'il contient, à la ligne 2. Le vecteur utilisé pour représenter le cluster utilisé pour le comparer aux vecteurs de tweets aux lignes 5 et 9 est le vecteur moyen de ceux des tweets du cluster. Les auteurs définissent un évènement comme « a real-world occurrence e with (1) an associated time period T_e and (2) a time-ordered stream of Twitter messages M_e, of substantial volume, discussing the occurrence and published during time T_e ». À intervalle de temps réguliers, les clusters qui ont agrégé plus de tweets qu'un seuil μ' sont identifiés comme relatifs à un évènement. Leurs caractéristiques sont passées dans un classifieur qui identifie la nature de l'évènement. Pour cela, Becker et al. proposent d'employer le taux de croissance horaire d'un cluster, la proportion de ses messages contenant des interactions sociales (mention, retweet et réponse) et les proportions de co-occurrences de ses mots.

Cette méthode permet de traiter et de grouper les messages au fur et à mesure de leur arrivée. Elle souffre toutefois du fait que les messages n'arrivent pas nécessairement dans un ordre optimal. Des clusters peuvent ainsi être initialisés avec des messages qui auraient été assignés à un autre cluster s'ils avaient été reçus à un autre moment. Des clusters de la partition pourraient alors être groupés. On parle de fragmentation de cluster.

Une solution à ce problème est présentée dans (Hasan et al., 2019). L'approche proposée est globalement similaire à celle de (Becker et al., 2011). Les tweets sont représentés avec des vecteurs de score TF-IDF et sont clusterisés en se basant sur la distance du cosinus. Elle diffère toutefois sur plusieurs points. Elle maintient un index inversé qui lie Q (paramètre fixé) mots au M (paramètre fixé) tweets les plus récents. Ces Q mots sont ceux avec les plus hauts scores TF-IDF de ces messages. Quand un nouveau tweet t est reçu, ses K (paramètre fixé) mots avec les plus hauts scores TF-IDF sont extraits et les messages récents contenant ces mots sont récupérés grâce à l'index inversé. t est comparé

à chacun de ces messages avec $dCOS$. Si cette distance est plus basse qu'un seuil paramétré, t est considéré comme "non nouveau" et est envoyé au module de clustering de l'approche. Sinon, il remplace le plus ancien des M tweets récents.

Le module de clustering fonctionne ensuite comme celui de (Hasan et al., 2019). Étant donné la partition courante $\mathcal{P}$, t est assigné au cluster c tel que $dCOS(\vec{t}, \vec{c}) < \mu$ et $\forall c' \in \mathcal{P},\ c' \neq c,\ dCOS(\vec{t}, \vec{c}) < dCOS(\vec{t}, \vec{c'})$. Ce module répond au problème de fragmentation de cluster en embarquant un autre seuil paramétré noté g_{ev}. Pour chaque test $dCOS(\vec{t}, \vec{c}) < \mu$ effectué, le module effectue également un test $dCOS(\vec{t}, \vec{c}) < \mu - g_{ev}$ et retient dans un ensemble S_c les clusters c qui le satisfont. Quand t est assigné au cluster qui minimise $dCOS(\vec{t}, \vec{c})$, c et les éventuels clusters dans S_c sont fusionnés. C'est ainsi que Hasan et al. proposent d'adresser le problème de fragmentation de cluster de leur approche incrémentale.

Ces exemples permettent d'illustrer comment un clustering incrémental plat est construit. À l'aide d'une mesure de similarité (ou réciproquement de distance), les documents reçus sont comparés entre eux. Cela permet de raccrocher le message à un cluster existant ou d'en créer un nouveau si aucun ne convient.

Il est également possible de mettre à profit le clustering hiérarchique dans un contexte incrémental.

2.3.3.2 Clustering incrémental hiérarchique

La structure arborescente des clusterings hiérarchiques permet de limiter le nombre de comparaisons à faire pour intégrer une nouvelle instance à la partition existante.

Dans (Sahoo et al., 2006), les auteurs proposent une adaptation du clustering incrémental hiérarchique *COBWEB* de (Fisher, 1987). Cet algorithme fonctionne sur des données transactionnelles, sur la base d'un critère de qualité de l'arbre obtenu appelé *Category Utility* ou CU. En calculant la CU de l'arbre avec et sans une transaction dans un des noeuds de l'arbre, on peut calculer le gain d'utilité apporté par l'attribution de cette transaction à ce noeud. Ainsi *COBWEB* peut identifier le noeud auquel rattacher chaque nouvelle transaction t pour maximiser la CU de l'arbre entier. Sachant le cluster lié à ce noeud, il décide s'il faut directement lui rattacher t, le diviser ou le fusionner avec le cluster du second meilleur noeud en CU. La décision prise est celle qui maximise la CU de l'arbre.

Sahoo et al. adaptent *COBWEB* au traitement de données textuelles en représentant les documents avec des vecteurs des décomptes des mots qu'ils contiennent. Chaque document est un vecteur $(a_1, ..., a_l)$ où l est la taille de lexique. Étant donné le nouveau document d à insérer dans l'arbre représentant cluster, la CU adaptée prend la forme :

$$CU_p(d) = \sum_k \frac{P(C_k)[\sum_{i=1}^{l} CU_{i,k} - \sum_{i=1}^{l} CU_{i,p}]}{K} \tag{2.6}$$

où $CU_p(d)$ est la CU calculée pour le noeud p, soit le cluster p, si on lui attribue d et qui s'exprime en fonction des K enfants de p ; $P(C_k)$ est la probabilité que le document évalué soit dans le k-ème enfant de p si on attribue ce document à p ; et $CU_{i,k}$ est la contribution du i-ème attribut de d à $CU_p(d)$ qui se calcule avec :

$$CU_{i,k}(d) = 1 - \frac{2 \times df \times (N - \frac{cf \times df}{2 \times cf - df})}{N^2} \tag{2.7}$$

où df est le nombre de documents du corpus dans lesquels le mot i apparaît, cf est le nombre d'occurrences du mot i dans le corpus, et N est le nombre de documents dans le corpus. Ces valeurs sont décomptées dans le cluster k. En appliquant le fonctionnement de *COBWEB*, cette *CU* adaptée permet de construire l'arbre représentant le clustering incrémentalement au fil de la réception des documents.

Cette approche illustre comment tenir à jour une arborescence de clustering hiérarchique dans le cadre incrémental. Étant donné un nouveau document d, on cherche le noeud auquel le rattacher de sorte à maximiser la qualité de l'intégralité de l'arbre, comme la *CU* dans (Sahoo et al., 2006). Quand ce noeud est identifié, on évalue s'il vaut mieux lui attribuer directement d ou modifier la structure de l'arbre à partir de d. L'arbre est ainsi construit dynamiquement au fil de l'arrivée des données.

Ce type d'approche hiérarchique peut être adapté pour s'appuyer sur des plongements neuronaux, comme nous souhaitons le faire pour travailler à partir de la sémantique de ces derniers. La mesure de qualité de l'arbre s'appuie alors sur les distances dans l'espace de représentation. Toutefois, en les adaptant de la sorte, ces approches finissent par avoir des objectifs similaires à ceux des clusterings plats minimisant les distances au sein des clusters. Elles sont plus coûteuses à faire fonctionner, car calculant la qualité pour tout l'arbre à chaque ajout de données, et aboutissent à des résultats similaires. Comme notre cas d'étude n'accorde pas d'intérêt particulier à une forme arborescente de clustering, nous n'employons pas ce genre d'approche.

2.3.3.3 Clustering incrémental conceptuel

Pour adapter le clustering conceptuel à la modalité incrémentale, il faut modifier la recherche des concepts qui permettent de déduire les clusters.

En exploitant les *Frequent Term-Sets* de (Beil et al., 2002), les auteurs de (Hennig and Wurst, 2006) proposent de traiter des flux d'articles de presse. Initialement, les *FTS* sont identifiés avec l'algorithme Apriori (Agrawal et al., 1994) et ne sont retenus que ceux qui ont un nombre d'occurrences supérieur à un seuil paramétré. Comme l'algorithme Apriori identifie de nombreux *FTS*, il faut sélectionner ceux qui sont pertinents. Pour ce faire, ils sont arrangés dans une structure arborescente où chaque noeud est un *FTS* dont la taille est la profondeur où il se situe. Pour sélectionner les *FTS* retenus, Hennig et Wurst proposent de minimiser la fonction :

$$GCost(F_i) = \frac{LCost(F_i)|cov(F_i)| + \sum_{F_j \in M(F_i)} GCost(F_j)|cov(F_j)|}{|cov(F_i)| + \sum_{F_j \in M(F_i), M(F_i) \neq \emptyset} |cov(F_j)|} \tag{2.8}$$

où F_i est le *FTS* évalué, $cov(F_i)$ est l'ensemble des documents qui contiennent tous les termes de F_i et $M(F_i)$ est l'ensemble des *FTS* successeurs de F_i, choisis dans l'ensemble $N(F_i)$ de tous les successeurs de F_i car optimisant la fonction de coût. *GCost* estime la qualité de tout l'arbre et est exprimée en fonction de *FCost* qui estime celle d'un cluster telle que :

$$FCost(F_i) = \sum_{d \in C(d_i)} (m_i(d) - 1) + \sum_{d \in cov(F_i) \setminus C(d_i)} n_i(d) \tag{2.9}$$

où $C(d_i)$ est l'ensemble des documents qui contiennent au moins des termes de F_i, $m_i(d)$ est le nombre de *FTS* de $M(F_i)$ qui couvre d et $n_i(d)$ est le nombre de *FTS* de $N(F_i)$ couvrant d et qui ne sont pas dans $M(F_i)$.

Une fois cette structure établie, on peut commencer à recevoir le flux de données. Chaque nouveau document d' est rattaché au cluster lié du FTS connu le plus long qu'il contient, ou stocké dans un "fourre-tout" jusqu'au prochain calcul des ensembles fréquents. Les documents dont la durée de vie est supérieure à une limite prédéfinie sont supprimés et au bout d'un certain nombre de nouveaux documents reçus, les FTS sont mis à jour. Ainsi un FTS ne couvrant plus assez de documents est supprimé de l'arborescence courante. Si un autre choix de structure permet de réduire la valeur $GCost$, les changements sont appliqués aux noeuds concernés dans l'arbre. Le nombre d'opérations à effectuer pour tenir l'arbre est ainsi maîtrisé.

Cet exemple illustre la façon d'adapter le clustering conceptuel au cadre incrémental. Quand un nouveau document d est reçu, on vérifie s'il est couvert par un concept connu. Si c'est le cas, d est attribué au cluster déduit de ce concept c. On vérifie alors que d déclenche la mise à jour de c. Cela peut consister à ajouter des éléments dans c, le diviser en plus petits concepts s'il devient trop long ou si cela permet d'identifier des concepts plus courts qui améliorent la qualité du clustering global. Sinon, si aucun concept connu ne couvre d, il est ajouté à l'ensemble nc des éléments qui ne sont couverts par aucun concept. Quand les concepts connus sont mis à jour, on vérifie si des éléments de nc peuvent être intégrés du fait de cette modification.

Toutefois, comme pour le clustering non incrémental, les méthodologies conceptuelles ne permettent pas de tirer parti des plongements sémantiques. Nous ne les retenons donc pas pour notre cas d'étude.

Chapitre 3

Description de données textuelles

Sommaire

Quand on travaille sur une grande quantité de données, il est nécessaire d'en produire un résumé ou une caractérisation pour permettre d'en imager la substance. Le volume d'un grand corpus le rend en effet impossible à appréhender pour un humain sans les outils adéquats. Lorsque les données en question sont textuelles, la description est un outil tout indiqué.

Dans le cadre de nos travaux il s'agit de décrire des ensembles de tweets, regroupés avec un algorithme de clustering sur la base de la sémantique capturée par le modèle de

plongement neuronal utilisé. Cette description doit donc permettre d'identifier ce dont parle le cluster.

Notre objectif est donc de décrire des clusters de documents, eux-mêmes composés de mots. Il est alors naturel de considérer les mots comme des éléments de description pour des ensembles de documents. « Keyword extraction is tasked with the automatic identification of a set of the terms that best describe the subject of a document » (Beliga et al., 2015). L'objectif est donc d'identifier les mots clefs à l'échelle des clusters de documents. Dans les données textuelles, et plus encore quand il s'agit de langue naturelle, tous les mots n'ont pas la même importance du fait de la construction de la langue ou d'usages propres au discours. Il faut donc identifier les mots ou les phrases les plus appropriés pour décrire les données traitées. Ils peuvent ainsi devenir les descripteurs, ou tags, constituant les descriptions des documents concernés.

Regrouper les documents pour les décrire n'est pas la seule approche qui existe dans la littérature pour décrire un ou des documents textuels. Nous présentons dans ce chapitre un panorama des méthodes existantes qui sont similaires à l'approche que nous développons.

Approche extractive contre approche abstractive Le résumé automatique se divise en deux grands axes de recherche. Les approches extractives visent à identifier des sous-parties, des mots ou des phrases, représentatives du ou des textes à résumer et à les extraire pour en présenter un aperçu. Les approches dites "abstractives" visent à reformuler et généraliser les données comme le ferait un humain pour les présenter, pour obtenir un résultat plus facile à exploiter. Cette transformation des données d'origine n'est pas utile dans le cadre de nos travaux. Notre objectif est de présenter le contenu issu de Twitter pour décrire un évènement et non de transformer ce contenu. Nous ne présentons donc pas ici les méthodologies abstractives.

3.1 Expliquer ou décrire ?

Rendre un résultat intelligible pour un humain est un domaine de recherche à part entière. On trouve par exemple des travaux poursuivant cet objectif en apprentissage automatique avec le domaine de l'explicabilité. Le faire pour un cluster peut tenir de l'explication ou de la description de ce résultat. Ces deux tâches sont similaires mais pas équivalentes. Leur définition est discutée dans plusieurs grands domaines de la recherche comme l'informatique dans le cas présent, mais également en philosophie, en mathématiques ou en sociologie. Il nous faut donc définir les objets dont nous parlons.

La figure 3.1 montre comment des méthodologies d'explications permettent d'expliquer la décision d'un modèle en montrant quelles parties d'une entrée de celui-ci sont à l'origine d'un résultat à sa sortie. Sur cette image, on voit des exemples d'explication de la tâche de reconnaissance d'objets dans le domaine de la vision par ordinateur. Si ce domaine est trop éloigné de nos travaux pour que nous le présentions, la tâche de reconnaissance d'objet permet toutefois de montrer efficacement la différence entre explication et description.

L'approche présentée est celle des cartes à chaleur. Une carte à chaleur met en évidence les parties d'une entrée qui ont eu le plus d'importance dans le calcul d'une sortie. Dans cet exemple, une même reconnaissance de borne d'incendie est expliquée avec quatre méthodes différentes pour montrer les différences dans les explications obtenues. Avec l'image en entrée du modèle à gauche, on voit sur les deux images à sa droite les parties

FIGURE 3.1 – Visualisations d'explications par carte à chaleur pour de la reconnaissance d'objet dans des images (Sun et al., 2022).

plus déterminantes pour la décision du modèle en rouge. Les pixels de l'image d'entrée menant à cette décision sont mis en évidence dans les deux dernières images de droite, en violet dans l'avant dernière et en rouge dans l'autre.

Une explication consiste à rendre intelligibles les mécanismes à l'origine d'un résultat. Formulé autrement, « an explanation says how it came to be like that » (Dowden, 1993). Elle peut même consister en un modèle simplifié plus accessible qui émule le comportement du modèle. C'est une tâche complexe qui implique de pouvoir détailler toutes les mécaniques qui sous-tendent son fonctionnement. Elle a notamment pris de l'ampleur du fait du besoin grandissant de pouvoir expliquer les résultats des modèles neuronaux profonds fonctionnant en boîtes noires. L'information apprise par les couches cachées des modèles n'est pas humainement interprétable. Plusieurs méthodologies ont vu le jour pour, par exemple, mettre en évidence les caractéristiques ou sous-parties des entrées à l'origine d'un résultat donné, comme celle proposée dans (Elguendouze et al., 2022). Dans le monde des modèles d'apprentissages neuronaux, c'est le champ de recherche de l'explicabilité, un domaine consistant en l'étude et la mise en évidence des liens entre les données et les décisions de ces modèles.

Dans la figure 3.1, l'explication prend la forme de la mise en évidence des parties d'une image qui ont mené le modèle à y identifier une borne d'incendie. Ainsi, on peut dire que si le modèle a identifié cet objet, c'est parce que l'image contient les pixels mis en évidence dans les quatre images de droite de la figure 3.1. De fait, on peut lier leur présence et leurs caractéristiques (couleur, position, luminosité, etc) à la détection d'une bouche d'incendie. Cela ne suffit toutefois pas à le décrire.

Décrire est une tâche différente, « [it] says that it's like that » (Dowden, 1993). L'objectif est plus direct et consiste à rendre état des caractéristiques des résultats obtenus, sans nécessairement s'intéresser à comment on les a obtenus. On peut alors considérer que la tâche de description pourrait être inclue dans la tâche d'explication. En effet, une explication détaillant le processus permettant d'obtenir un résultat pourrait aussi permettre de le décrire. Toutefois, une explication peut ne pas suffire pour décrire un résultat. Pour en revenir aux explications de détection de bouche d'incendie en figure 3.1, celles-ci pourraient permettre de la décrire avec sa couleur et sa forme, qui feraient partie de l'explication. Mais cela n'aurait pas besoin d'être relié à une décision du modèle. L'objet sur l'image a cette forme et cette couleur, peu importe quel est l'objet reconnu.

Notons que les tâches de descriptions et d'explications dépendent de l'usage qu'on en fait. Si on n'a pas besoin de savoir à quoi sert une bouche d'incendie, l'explication avec sa couleur et sa forme suffit à le décrire. Si on a plutôt besoin de savoir à quoi sert cette bouche d'incendie, la description doit intégrer cette information.

Une description peut rarement être considérée comme une explication. Sur l'exemple de la figure 3.1, savoir qu'une borne d'incendie est une source d'eau ne permettrait pas au modèle de la détecter dans l'image. Cela n'apporterait aucune information sur le fonctionnement de ce modèle. Si cela peut donc être un élément important de la description, ce n'est pas un élément d'explication.

La sélection de ces éléments appelés descripteurs, ou tags, représente un point de difficulté pour une explication comme pour une description. Leur qualité dépendra mécaniquement de la qualité des tags utilisés pour les constituer. Dans le cas d'une explication, cela peut consister en une série de formules permettant le passage d'une entrée à une sortie. Cela peut aussi consister en une liste d'attributs qui justifie l'étiquetage d'une instance dans une classe donnée. Les descriptions prennent d'ailleurs toujours cette forme, d'où la nécessité de choisir des descripteurs de bonne qualité. Un point à garder à l'esprit est que l'importance d'un descripteur est à mettre en perspective avec la tâche traitée. Pour terminer avec l'exemple de la figure 3.1, si l'objectif est de décrire cette borne d'incendie, faut-il rendre compte de son usage, ou dire qu'elle est rouge et située sur une plate-bande d'herbe suffit-il ? Cette question ne peut trouver de réponse que par rapport à l'utilisation qui sera faite de la description.

Dans notre cas, étant donné un ensemble de documents, l'objectif est de le décrire avec des mots que ceux-ci contiennent. Ces mots doivent être représentatifs de l'ensemble décrit et permettre d'en imager le contenu.

3.2 Approches sans partitionnement

Nous présentons dans cette section les méthodologies qui permettent de décrire un ou plusieurs documents, sans mobiliser de clustering pour détecter une structure sous-jacente.

3.2.1 Approches à base de score

Une manière répandue d'identifier les mots et les phrases à extraire des données textuelles pour les décrire est de leur attribuer un score. Cela permet de les classer et de retenir les meilleurs pour identifier les plus pertinents à sélectionner, au regard du critère mesuré. Nous présentons dans cette sous-section les différents types de scores qui peuvent être mis à profit.

3.2.1.1 TF-IDF

Le score TF-IDF présenté en section 1.2.3 permet de mesurer numériquement à quel point un mot distingue son document des autres du même corpus. Rappelons ici sa formule.

$$TF - IDF(i, d, D) = f_{i,d}.log(\frac{|D|}{|\{d' \in D : i \in d'\}|}) \tag{3.1}$$

Comme on le voit dans l'équation 3.1, un mot w obtient le score TF-IDF maximal qu'il peut atteindre dans le cas théorique où le document d dans lequel il apparaît consiste uniquement en une répétition infinie de w, et que d est le seul document du corpus à contenir w.

Comme l'opérande IDF fait chuter le score des mots fréquents dans tout le corpus d'étude, cet outil permet de repérer efficacement les mots "outils" de la langue traitée. Ces derniers apparaissent de manière uniforme car ils ne sont pas spécifiques à une information particulière. Ils permettent de construire la langue et peuvent ainsi apparaître dans n'importe quel contexte. Réciproquement, les mots excessivement rares car n'apparaissant que quelques fois dans leur corpus, auront un score bas du fait de l'opérande TF. Cela permet d'isoler les mots trop rares pour être significatifs.

Ainsi le TF-IDF est un outil standard pour extraire des mots saillants d'un ou de plusieurs documents, au regard de la collection de documents considérée. Par exemple, dans (Liu et al., 2018) et (Zhao et al., 2018), les meilleurs scores TF-IDF identifient les mots utilisés pour décrire les documents. Le TF-IDF est le principal représentant des scores basés sur la fréquence des mots dans leur document. Les informations statistiques à propos des mots et des phrases sont simples à calculer et peuvent être mesurées dans presque tous les types de données. Elles permettent d'attaquer facilement les données pour en tirer du sens.

Toutefois, dans le cadre de nos travaux, ce score présente une limitation majeure. Sa partie TF, qui compte le nombre d'occurrences du mot au sein d'un document, n'est pas adaptée aux documents courts. Un message qui ne peut pas comporter plus de 280 caractères ne contient généralement pas de répétition en dehors des mots outils, comme "le" ou "la" en français. Malgré cette limitation, cette méthodologie à base de fréquence est puissante pour traiter des données au sein d'ensembles, ce qui correspond à notre utilisation du clustering. Nous proposons donc d'adapter le TF-IDF, comme nous le verrons en section 5.1.2.2.

3.2.1.2 Scores sur caractéristique sociale

Décrire peut consister en l'extraction de phrases clefs. Une façon d'aborder le problème est de considérer ces phrases comme des combinaisons de mots clefs. Ainsi dans (Zhao et al., 2011a), les auteurs proposent de commencer par chercher des mots clefs avec le modèle thématique Twitter-LDA présenté en section 3.2.3. Le modèle est utilisé pour identifier les sujets des tweets et leurs mots clefs. Les mots sont ensuite représentés dans un graphe où deux sommets reliés par un arc représentent deux mots qui co-occurrent dans des tweets assignés au même sujet. L'arc représente la probabilité d'apparition d'un mot dans un message connaissant l'autre. Ce graphe permet ainsi d'identifier des phrases composées des mots clefs les plus susceptibles d'apparaître ensemble étant donné un sujet. On obtient ainsi des candidats de phrases clefs, qui sont classés selon le score suivant.

$$Score_t(p) = \log \frac{|ReTweets_p| + 1.0}{|Tweets_p| + l_{avg}} + (\sum_{w \in p} \log \frac{\#(C_t, w) + \beta}{\#(C, w) + \mu})$$

Ce score est calculé pour le candidat de phrase clef p par rapport au sujet t. $ReTweets_p$ et $Tweets_p$ sont respectivement les ensembles de retweets et de tweets contenant p ; l_{avg} est une constante calculée à l'échelle du corpus et est le nombre moyen de tweets dans lesquels une phrase clef apparaît ; $\#(C_t, w)$ et $\#(C_t, w)$ sont respectivement les nombres de fois où le mot w apparaît dans les messages du sujet t et dans tous les messages du corpus ; β et μ sont des paramètres à fixer empiriquement pour la distribution de la thématique de fond du corpus (voir section 3.2.3). Finalement, le résultat est composé des meilleures

phrases selon ce score.

Les scores basés sur les caractéristiques "sociales" d'un tweet, comme son nombre de retweets ou les communautés dans lesquelles il est publié, ne retiennent pas notre intérêt. Ce sont des mesures qui sont utiles pour caractériser la diffusion des messages et de l'information qu'ils contiennent. Notre objectif est de décrire un évènement à travers le contenu publié à son sujet. Nous mesurons l'importance de ses aspects à la variété de messages originaux qui les mentionnent. Si beaucoup d'utilisateurs différents en parlent, ces aspects seront représentés par les clusters que nous construisons. Ainsi, nous n'employons pas les caractéristiques sociales des données pour attribuer un score aux éléments des documents.

Les statistiques ne sont pas la seule base de travail pour établir des scores. En travaillant à partir d'une représentation en espace vectoriel, un score peut être calculé à partir de la similarité sémantique mesurée dans cet espace.

3.2.2 Graphes et mesure de similarité

Représenter les données textuelles sous forme de graphe permet d'y appliquer des raisonnements issus de ce domaine. Un graphe $G = (V, E)$ est une paire composée d'un ensemble de sommets et d'un ensemble de paires de sommets, appelées arêtes ou arcs, représentant des liens entre ces sommets. Le graphe peut être orienté si les arêtes ont un sens, c'est-à-dire que l'arête (a, b) est différente de l'arête (b, a). Les arêtes peuvent également être pondérées.

On peut ainsi représenter un ensemble de documents avec un graphe dont les sommets sont les mots du corpus et les arêtes représentent les co-occurrences. Si deux mots apparaissent dans un même contexte (voisinage, phrase ou document selon les approches), le graphe comporte une arête entre les deux sommets représentant ces mots.

Différentes métriques de graphes peuvent alors être employées pour étudier les mots. En particulier, le degré d'un sommet est beaucoup utilisé. Pour un sommet v, c'est le nombre de ses voisins soit le nombre d'arêtes qui contiennent v :

$$deg(v) = |\{e \in E : v \in e\}|$$

Dans (Boudin, 2013), l'auteur aborde un type de métrique de graphe appelé centralité dans le but d'identifier les mots pour décrire un ou plusieurs documents. Plusieurs variantes de cette mesure existent, comme celle de degré qui est le rapport entre le degré d'un sommet et le nombre total de sommets du graphe, tel que :

$$c_d(v) = \frac{deg(v)}{|V| - 1}$$

Boudin compare différents calculs de centralité pour évaluer lesquels sont les plus appropriés pour identifier des descripteurs de données textuelles. Il montre en particulier que cette métrique permet d'obtenir de meilleurs résultats que des métriques statistiques comme le TF-IDF.

Les représentations par graphe construites avec ce type d'approche se basent sur les statistiques au sein des données, sans permettre de s'appuyer sur leur sémantique. Pour contourner ce problème, on peut baser le graphe sur des plongements neuronaux pour mettre à profit la similarité sémantique qu'ils permettent de mesurer.

Pour exploiter les modèles récents, les auteurs de (Liang et al., 2021) proposent d'exploiter des plongements du modèle BERT (Devlin et al., 2018), un modèle neuronal profond basé sur l'architecture du transformer. L'objectif de ces travaux est d'extraire des phrases significatives pour résumer un article de presse. Les modèles basés sur l'architecture des transformers construisent des plongements dits dynamiques, par rapport à ceux statiques appris par un modèle comme Word2Vec. Un vecteur de mot dans cet espace est spécifique à ce mot dans un contexte particulier, ce qui permet d'avoir plusieurs vecteurs pour un même mot. Ainsi, le mot "tempête" sera représenté avec un vecteur pour son apparition dans "La solitude est une tempête de silence" et avec un autre vecteur pour "Solidaires avec les sinistrés de la tempête". Cela permet par exemple de distinguer les différents sens d'une homonymie. Les modèles comme BERT sont de plus entraînés sur de très grands ensembles de données pour capturer une information générale sur la langue, pour reconnaître le plus d'usages différents des mots possibles.

Liang et al. utilisent ce large modèle linguistique pour inférer les représentations des mots du document étudié. Les représentations des phrases candidates sont calculées en faisant la moyenne de celles de leurs mots. Une phrase du document étudié est retenue comme candidate pour le résumer s'il s'agit d'une phrase nominale composée d'au moins un nom, avec ou sans adjectif. Les candidates sont utilisées comme sommets V d'un graphe $G = (V, E)$ où les arcs E représentent la similarité entre deux phrases candidates. Cette similarité est calculée avec :

$$\texttt{sim}(H_i, H_j) = \vec{H_i}^{\intercal} \cdot \vec{H_j}$$

où H_i et H_j sont les i-ème et j-ème phrases du document.

Cette représentation en graphe permet de calculer, pour une phrase candidate, sa centralité avec :

$$C(H_i) = \sum_{j=1, j\neq i}^{|V|} \texttt{sim}(H_i, H_j)$$

On voit ici que l'espace de plongement des données textuelles permet de baser le calcul de centralité sur la similarité sémantique. Dans ces travaux, les auteurs proposent de tenir compte de la position de la phrase dans le document dans le calcul de son importance. À cette fin, ils modifient le calcul de centralité d'un sommet du graphe en :

$$C(H_i) = \sum_{d_b(i)<d_b(j)} \texttt{sim}(H_i, H_j) + \lambda \sum_{d_b(i)\geq d_b(j)} \texttt{sim}(H_i, H_j)$$

où $d_b(i)$ est une fonction proposée par les auteurs telle que $d_b(i) = min(i, \alpha(n-i))$, avec n la longueur du document étudié.

Cette fonction permet de déterminer si la position d'une phrase la place au début ou à la fin du document. Comme les phrases du début des articles de presse sont généralement les plus importantes pour les résumer, Liang et al. proposent d'intégrer cette information au calcul de centralité. Le paramètre λ, inférieur à 1, limite l'apport à la valeur de centralité de la phrase H_i venant d'une phrase candidate H_j plus proche du centre du document. $C(H_i)$ est considérée comme la partie locale du score à attribuer à H_i pour décider s'il faut la retenir pour décrire le document.

La partie globale de ce score, notée $R(H_i)$, est calculée en comparant la représentation de la phrase à celle du document dont elle est issue. Cette dernière est calculée en appliquant un maxpooling sur les mots du document :

$$\vec{d} = \texttt{Maxpooling}_t(w|w \in d)$$

Notons que $\vec{H_i}$ et $\vec{d}$ sont de même dimension. Le maxpooling est une opération permettant de mettre en avant les valeurs maximales dans des vecteurs. Ainsi, les vecteurs de documents sont constitués avec les valeurs les plus élevées tirées des vecteurs de leur mot.

La représentation d'une phrase candidate est comparée à celle du document avec :

$$R(H_i) = \frac{1}{dMAN(\vec{d}, \vec{H_i})}$$

où $dMAN$ est la distance de Manhattan présentée en section 2.1.2.

Les phrases sont classées selon leur score S, calculé comme la combinaison des scores globaux et locaux des phrases avec :

$$S(H_i) = R(H_i) \cdot C(H_i)$$

Les phrases ayant le plus haut score S sont extraites pour résumer le document dont elles sont issues.

Cette approche en particulier n'est pas applicable dans le cadre de nos travaux puisqu'elle est pensée et construite pour traiter des articles de presse. Si la question de la position d'une phrase dans ce type document est importante, elle ne se pose pas dans un message aussi court qu'un tweet. De plus, nous faisons le choix de ne pas employer de modèle linguistique de très grande taille pour calculer des représentations des mots et des documents. Comme notre objectif est de construire des représentations afin de grouper les documents, nous n'avons pas besoin d'une ressource linguistique aussi imposante. Un modèle capturant la sémantique représentée dans les données étudiées est suffisant. Par ailleurs, nous voyons dans (Liang et al., 2021) que la représentation du document doit être calculée à partir des représentations des mots. C'est une étape intermédiaire supplémentaire par rapport à un modèle qui apprend directement des vecteurs pour les documents. En plus de complexifier le calcul de ces représentations, cela peut compliquer la compréhension d'un système basé sur ces représentations.

Plus généralement, les approches de résumé ou description à base de graphes ne correspondent pas à l'approche que nous souhaitons mettre en place. Pour pouvoir s'appuyer sur la sémantique des documents à décrire pour construire un graphe permettant de les décrire, il faut en calculer des représentations dans un espace de plongement. Une fois ce dernier calculé, le clustering travaille directement sur sa topographie pour identifier des ensembles de documents proches, là où l'approche par graphe doit d'abord construire celui-ci. Nous optons donc pour l'approche par partitionnement, pour nous appuyer sur la détection d'une structure sous-jacente organisant les données.

3.2.3 Modèles thématiques

Aussi appelés modèles de sujet, les « topic models » sont des modèles statistiques dont l'objectif est de capturer les sujets latents cachés dans des ensembles de documents.

La tâche de modélisation de sujet considère qu'un document contient un mélange de thématiques, identifiable à travers les distributions des mots au sein des documents. Ainsi, l'objectif des modèles de sujet est d'identifier les distributions de mots qui caractérisent les sujets au sein des documents d'un corpus.

L'intuition derrière le fonctionnement de ces méthodologies est que les mots employés pour parler d'un même sujet ont plus de chances d'apparaître ensemble. Ainsi, au moment d'écrire un énoncé sur une thématique, un locuteur utilise certains mots qui amènent l'utilisation d'autres mots liés à cette thématique. L'utilisation de ces mots suivrait alors une distribution de probabilité d'apparition propre à cette thématique. Ce sont ces distributions que cherchent à estimer les modèles de sujet et il existe différentes méthodologies pour inférer ces distributions.

La plus répandue est l'allocation de Dirichlet latente, appelée *LDA* (« *Latent Dirichlet Allocation* ») proposée initialement dans (Pritchard et al., 2000), adaptée aux corpus de texte dans (Blei, 2003) et toujours utilisée actuellement comme dans (Gurcan et al., 2021). Soient un corpus D composé de M documents, N_i le nombre de mots dans le i-ème document de D, avec α et β les vecteurs paramètres des lois de Dirichlet pour les distributions des sujets par document et pour la distribution de sujet par mot. Notons θ_i la distribution de sujet pour le document i et φ_k la distribution de mots pour le sujet k, avec K le nombre de sujets à découvrir dans les données.

Avant de présenter l'algorithme de la LDA, rappelons les définitions des lois de probabilités sur lesquelles cet algorithme s'appuie. En notant $X_1, X_2, ..., X_n$ des variables aléatoires de Bernoulli suivant la loi binomiale de paramètre p, indépendantes et aléatoires, leur somme N est une variable aléatoire suivant la loi binomiale et s'écrit :

$$N = \sum_{k=1}^{n} X_k \sim \mathcal{B}(n, p)$$

avec $\mathcal{B}$ la loi binomiale de fonction de masse $\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$.

La loi multinomiale de paramètre p concerne les cas où les tirages dans la distribution sont répétés n fois et peuvent prendre m valeurs différentes. En notant N_i le nombre de fois où la variable n obtient la valeur i, la loi multinomiale s'écrit :

$$\mathbb{P}(N_1 = n_1, ..., N_m = n_m) = \binom{n}{n_1, n_2, ..., n_m} p_1^{n_1} ... p_m^{n_m}$$

Avec un vecteur de réels positifs α comme paramètre, la loi de Dirichlet d'ordre $K \geq 2$ s'écrit :

$$f(x_1, ..., x_K; \alpha_1, ..., \alpha_K) = \frac{1}{B(\alpha)} \prod_{i_1}^{K} x_i^{\alpha_i - 1}$$

où les x_i sont des variables aléatoires multinomiales.

$B(\alpha)$ est la fonction bêta multinomiale, s'écrivant :

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}$$

où Γ est la fonction gamma telle que $\Gamma(n) = (n-1)!$

Algorithme 7: Fonctionnement de la LDA

input: corpus D, paramètres de loi de Dirichlet α et β

Pour chaque *document i dans D* **faire**
  tirer $\theta_i \sim Dir(\alpha)$
Pour chaque *sujet k* **faire**
  tirer $\varphi_k \sim Dir(\beta)$
Pour chaque *document i dans D* **faire**
  Pour chaque *mot j dans i* **faire**
    tirer $z_{ij} \sim Multi(\theta_i)$
    tirer $w_{ij} \sim Multi(\varphi_{z_{ij}})$

La LDA fonctionne de manière générative en suivant l'algorithme 7. Cet algorithme est répété jusqu'à ce que les paramètres α et β des lois de Dirichlet permettent d'obtenir des distributions correspondant aux données étudiées.

Dans l'algorithme 7, $Multi(x)$ est la distribution catégorielle ou multi-Bernoulli. L'apprentissage de toutes les distributions de probabilités présentes dans ce processus génératif est un problème d'inférence statistique. Ce dernier peut être résolu de différentes manières, comme l'inférence bayésienne ou l'échantillonnage de Gibbs. Nous ne les présentons pas ici car ces outils ne relèvent pas de notre sujet d'étude.

La modélisation thématique a été vastement explorée depuis sa proposition, comme on peut le voir dans (Chauhan and Shah, 2021), et a obtenu de bons résultats dans différents domaines d'application. Elle a toutefois montré des limitations pour traiter des corpus de documents courts, comme nous avons besoin de le faire dans le cadre de nos travaux. La principale cause de cette limitation vient du fait que dans des documents courts, l'utilisation du vocabulaire global du corpus est très éparse (Yan et al., 2013). Dans un corpus de documents classiques, le vocabulaire de chaque document est centré sur la thématique de celui-ci, de sorte à ce que le document soit cohérent de bout en bout. Dans un corpus de documents courts en revanche, le vocabulaire de chaque document se chevauche car le cadre imposé par la cohérence du document est bien plus restreint.

Pour contourner cet écueil, différentes adaptations ont été proposées. Avec (Zhao et al., 2011b), les auteurs proposent de modifier le processus génératif de LDA en y incorporant deux modifications. D'abord, une distribution pour une thématique de fond, commune à tous les tweets d'un corpus, est calculée en plus des autres distributions. C'est celle des mots qui ne sont raccrochés à aucun sujet clairement identifiable. Ensuite, les distributions de sujets sont calculées par tweet et par utilisateur, pour représenter le fait qu'un tweet est à propos d'un seul sujet et que celui-ci est lié à son auteur. Ces modifications sont présentées dans l'algorithme 8.

L'adaptation de la LDA pour traiter des tweets telle que présentée dans l'algorithme 8 permet de s'adapter aux mots qu'on ne peut attribuer à aucun sujet. C'est notamment le cas pour des mots très rares, propres à des conversations entre certains individus, ou encore pour des hashtags peu utilisés. De plus, le fait de calculer une distribution θ^u par utilisateur dans les données permet de s'adapter au contexte particulier du traitement de données Twitter. Les messages sont courts en plus d'être très nombreux et pour qu'un ensemble de messages soit un minimum cohérent, ils doivent être liés par une caractéristique commune.

Algorithme 8: Fonctionnement de la Twitter-LDA

input: corpus D, paramètres de loi de Dirichlet α, β et γ

tirer $\varphi^{\mathcal{B}} \sim Dir(\beta)$ `/* ` $\varphi^{\mathcal{B}}$ `est la distribution des mots de la thématique de fond */`

tirer $\pi \sim Dir(\gamma)$ `/* ` π `est une distribution de Bernoulli */`

Pour chaque *sujet* k **faire**

  tirer $\varphi_k \sim Dir(\beta)$

Pour chaque *utilisateur* $u = 1, ..., U$ **faire**

  tirer $\theta^u \sim Dir(\alpha)$ `/* ` θ^u `est la distribution des sujets pour l'utilisateur` u `*/`

  Pour chaque *tweet* $s = 1, ..., N_u$ *du corpus dont l'auteur est* u **faire**

    tirer $z_u \sim Multi(\theta^u)$

    Pour chaque *position de mot* $n = 1, ..., N_s$ *de* s **faire**

      `/* ` N_s `est la taille du tweet` s `*/`

      tirer $y \sim Multi(\pi)$ `/* ` y `est une valeur binaire qui représente le fait que` w `soit un mot de la thématique de fond ou pas */`

      Si $y = 1$ **alors**

        tirer $w_{ij} \sim Multi(\varphi^{\mathcal{B}})$

      sinon

        tirer $w_{ij} \sim Multi(\varphi^{z_{us}})$

Sans cette cohérence minimum, le topic modeling ne peut pas identifier de sujet. Ici, Zhao et al. ont choisi les auteurs des tweets comme trait commun. Cela permet de limiter le problème de l'utilisation éparse du vocabulaire dans le corpus de tweets. Dans (Sasaki et al., 2014), Sasaki et al. proposent d'améliorer la Twitter-LDA en considérant que la distribution π doit être calculée pour chaque utilisateur, de sorte à capturer le bruit de fond de Twitter de manière plus fine.

Dans (Yan et al., 2013), une autre façon d'adapter la LDA aux messages courts de Twitter est explorée. Le modèle proposé, appelé *BTM* pour "*Biterm Topic Model*", fonctionne essentiellement de la même manière que la LDA classique présentée dans l'algorithme 7. L'adaptation consiste à tirer une distribution des sujets pour tout le corpus d'étude, puis à tirer les distributions par bi-terme plutôt que par mots pour chaque sujet. Cette adaptation mène donc à l'algorithme 9.

Algorithme 9: Fonctionnement du modèle thématique bi-terme

input: corpus D, paramètres de loi de Dirichlet α et β

Pour chaque *sujet* k **faire**
  choisir $\varphi_k \sim Dir(\beta)$
tirer $\theta \sim Dir(\alpha)$, commune à tous les documents de D
Pour chaque *bi-terme* b *de l'ensemble* B *contenant tous les bi-termes de* D **faire**
  tirer $z_b \sim Multi(\theta)$
  tirer $w_{b1} \sim Multi(\varphi_{z_b})$ et $w_{b2} \sim Multi(\varphi_{z_b})$ `/*` w_{b1} `et` w_{b1} `sont les deux mots composants le bi-terme` b `*/`

Dans l'algorithme 9, l'ensemble B est composé de tous les bi-grammes du corpus, soit les paires de mots apparaissant successivement dans les données. Le modèle ne considère donc pas des mots mais uniquement des bi-grammes. Ce changement d'approche permet de considérer à nouveau que chaque tweet est composé d'un mélange de sujets comme avec la LDA classique, à l'inverse de la Twitter-LDA qui considère un sujet par tweet. En tirant les distributions par bi-terme plutôt que par mot, cette approche contourne le problème posé au modèle de sujet par l'utilisation éparse du vocabulaire du corpus. En effet, considérer les bi-termes permet d'étendre la taille des unités permettant de caractériser le sujet. Dans le même temps, une co-occurrence de bi-termes est mécaniquement plus rare qu'une co-occurrence de mots et a donc plus de chances d'être caractéristique de son sujet.

La LDA, comme les autres approches de modélisation de sujet, produit des distributions de probabilités sur des mots par rapport à des sujets, on peut directement les représenter avec des vecteurs. Un de leurs emplois est donc de construire des représentations vectorielles des documents traités. Celles-ci représentent la sémantique des documents telle que capturée par le modèle de sujet employé. En s'appuyant sur cela et sur les représentations de mots telles que calculées par Word2Vec, présentées en section 1.3.4.1, (Moody, 2016) propose de construire des représentations vectorielles de mots tenant compte de leur document. Pour ce faire, les auteurs proposent de sommer les vecteurs de mots du document dans lequel ils apparaissent avec ceux calculés avec la LDA. Cette approche est appelée *lda2vec*. Cette démarche est adaptée à Twitter dans (Culmer and Uhlmann, 2021), où le problème de la LDA avec les messages courts est contourné en agglomérant les tweets selon leur auteur et leur date de parution. Le modèle proposé,

appelé *LDA2Vec-Tweet Time Pooling*, permet de construire des vecteurs de tweets et de mots qui capturent spécifiquement la sémantique exprimée par une source donnée dans une fenêtre de temps donnée.

Les modèles thématiques permettent d'identifier des mots clefs pertinents pour les sujets qu'ils découvrent mais sont toutefois coûteux à utiliser. L'apprentissage des distributions de probabilité et de leurs paramètres pour les sujets et leurs mots nécessite des ressources et du temps de calcul. Par exemple, le modèle BTM a besoin de plus de 13 heures pour traiter un corpus de 20 000 tweets et y détecter les mots clefs pour 10 sujets. À titre de comparaison, sur le même jeu de données, une approche à base de TF-IDF obtient des résultats moins bons mais s'exécute en environ deux minutes à puissance de calcul égale. Par ailleurs, le fait d'extraire des mots seuls pour décrire les sujets peut mener à des ambiguïtés. Par exemple, si le verbe "faire" fait partie des mots d'un sujet, il sera impossible de lui donner un sens.

L'apprentissage profond peut également être mis à profit pour identifier, décrire ou résumer, un ou des documents.

3.2.4 Apprentissage profond

Dans (Zhang et al., 2016), les auteurs proposent une architecture de modèle neuronal récurrent profond conçue pour extraire des mots et des phrases clefs en même temps, appelée "joint-layer RNN". Elle se compose d'une couche x d'entrée et de deux couches cachées h^1 et h^2. Ces deux couches cachées ont une couche de sortie chacune, $\hat{y}^1$ et $\hat{y}^2$. Notons que $\hat{y}^1$ a deux sorties où transite le même résultat, une qui sert de sortie à la couche et une qui alimente h^2 comme l'entrée de cette dernière. $\hat{y}^1$ et $\hat{y}^2$ sont les deux composantes de l'objectif du modèle. $\hat{y}^1$ indique si un mot w du document traité est un mot clef à retenir pour le décrire et $\hat{y}^2$ indique si ce mot est seul ou sa position dans une phrase clef ("début", "milieu" ou "fin").

Comme le modèle est récurrent, chaque étape de calcul au sein des couches cachées et de sortie est basée sur l'état du modèle à l'étape précédente.

Les fonctions d'activations des couches cachées au temps (ou étape) t sont :

$$h_t^1 = \sigma(\mathbf{W}^1 x_t, \mathbf{U}^1 h_{t-1}^1)$$
$$h_t^2 = \sigma(\mathbf{W}^2 h_t^1, \mathbf{U}^2 h_{t-1}^2)$$

où $\mathbf{W}^1$ et $\mathbf{W}^2$ sont les matrices de poids des données issues de la couche précédente pour h^1 et h^2, et $\mathbf{U}^1$ et $\mathbf{U}^2$ les matrices de poids pour l'état de la couche au temps $t-1$ pour h^1 et h^2.

Les fonctions des couches de sorties sont :

$$\hat{y}_t^1 = softmax(\mathbf{V}^1 h_t^1)$$

$$\hat{y}_t^2 = softmax(\mathbf{V}^2 h_t^2)$$

où $\mathbf{V}^1$ et $\mathbf{V}^2$ sont les matrices de poids pour les couches cachées h^1 et h^2.

Tous les paramètres du modèle sont regroupés dans un ensemble θ tel que :

$$\theta = \{\mathbf{X}, \mathbf{W}^1, \mathbf{W}^2, \mathbf{U}^1, \mathbf{U}^2, \mathbf{V}^1, \mathbf{V}^2\}$$

où $\mathbf{X}$ est la matrice de représentation des mots présentée au modèle.

L'objectif final du modèle est :

$$J(\theta) = \alpha J^1(\theta) + (1 - \alpha) J^2(\theta)$$

où $\alpha \in [0, 1]$ est un paramètre pondérant l'un par rapport à l'autre les objectifs d'identification en tant que mot clef et de détection de position dans une phrase clef. J^1 et J^2 sont les objectifs des couches $\hat{y}^1$ et $\hat{y}^2$ tels que :

$$J^1(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} dist(\hat{y}_t^1, y_t^1)$$

$$J^2(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} dist(\hat{y}_t^2, y_t^2)$$

où *dist* peut par exemple être la distance euclidienne. Le modèle s'entraîne sur N séquences d'entraînement (soit sur N documents) formant l'ensemble $D = \{(x_t, y_t^1, y_t^2)_{t=1}^{T_n}\}_{n=1}^{N}$. T_n est la taille du document n et t est la position du mot x dans le document n. Chaque x_t est ainsi annoté avec deux étiquettes, l'une indiquant s'il s'agit d'un mot clef, l'autre sa position dans une phrase clef le cas échéant.

Cette approche à architecture profonde propose de détecter en même temps mots et phrases clefs en joignant les objectifs servant à les reconnaître. Les phrases clefs ainsi détectées sont donc nécessairement des combinaisons de mots clefs. En effet, l'objectif identifiant la position du mot dans la phrase clef est construit par-dessus l'objectif de reconnaissance des mots. Du fait de la complexité de ses couches, le modèle nécessite un volume important de données pour s'entraîner. Qui plus est, chaque mot de ces données doit avoir deux étiquettes, une pour chaque objectif. Pour répondre à ce besoin, Zhang et al. proposent de collecter des tweets contenant des hashtags. Ces derniers deviennent les mots clefs de référence. Ceux qui sont composés, comme `"#hommagealabravoure"`, deviennent des phrases clefs de référence.

Le coût de mise en oeuvre en volume de données et temps d'entraînement de ce type d'approche nous amène à ne pas les mobiliser. Par ailleurs, la méthodologie de constitution des références pour l'entraînement du modèle implique qu'un mot clef ne pourra être reconnu comme tel que s'il existe en temps que hashtag. Nous ne souhaitons pas nous réduire à ces cas de figure. En effet, nous souhaitons pouvoir identifier n'importe quel mot du contenu d'un tweet comme descripteur du cluster auquel il appartient si cela est pertinent.

Les approches à base de réseaux neuronaux profonds ou de sujets thématiques sont coûteuses à entraîner ou à mettre en oeuvre, du fait de leur coût computationnel à l'exécution ou de la quantité de données annotées nécessaire à leur entraînement. Dans les cas où ces ressources d'entraînement n'existent pas, elles ne sont pas applicables.

3.3 Approches à base de clustering

Dans le cadre multi-document, agglomérer des ensembles de documents cohérents permet d'identifier une structure aidant à construire le résumé. Nous faisons ici un panorama des approches basées sur cette méthodologie dans laquelle s'inscrivent nos travaux.

3.3.1 Construire simultanément clusters et descriptions

Lorsque l'on souhaite décrire un ensemble de données partitionné, il existe deux possibilités. On ne peut en effet pas construire de description avant de savoir ce que l'on décrit. Soit on construit les clusters et leur description en même temps, soit on commence par identifier les clusters pour les décrire dans un second temps.

Nous commençons par présenter la première de ces approches.

3.3.1.1 Clustering conceptuel

Le clustering conceptuel, comme nous l'avons présenté en section 2.2, consiste à identifier des concepts qui permettent de grouper les instances dans les attributs de ces dernières. Si cette méthodologie pouvait être appliquée à des données textuelles, elle répondrait alors à la tâche de leur description. Les concepts identifiés constitueraient en effet les descriptions des clusters. Ce type d'approche présenterait l'avantage de construire les clusters et leur description en même temps.

Il est possible de considérer un document comme une transaction et ses mots comme les items de cette transaction. Cela permet de représenter un ensemble de documents comme une base de données transactionnelle et il devient possible d'y appliquer du clustering conceptuel. Pour éviter une explosion du coût de calcul comme avec les approches de Michalski présentées en section 2.2.1, il faut éviter d'avoir à comparer tous les attributs de tous les concepts pour identifier le clustering.

Pour contourner ce problème, de nombreux travaux exploitent les ensembles de termes fréquents. Ces derniers, souvent abrégés dans la littérature en FT pour "*Frequent Termsets*", sont des ensembles de mots qui apparaissent souvent dans un même contexte, dont la taille varie d'une approche à l'autre. Le nombre de fois qu'un ensemble doit apparaître pour être considéré fréquent dépend également des approches. Toutefois en langue naturelle, ces FT peuvent être très nombreux et il peut de plus être difficile de définir a priori un seuil d'occurrences fiable à partir duquel un ensemble de mots devient un FT. Pour éviter ce problème, les ensembles de termes fréquents sont filtrés pour ne retenir que ceux ayant la particularité d'être clos, appelés CFT pour *Closed Frequent Term-set*. La propriété de clôture d'un ensemble de termes se définit avec les opérateurs d'extension ext et d'intention int.

Définition 3.3.1 (Propriété de clôture)**.** Soient une base de données transactionnelle, avec $\mathcal{T}$ son ensemble de transactions, $\mathcal{I}$ celui des items et $cov(t, it)$ la relation indiquant que l'item it est présent dans la transaction t. Avec $A \subseteq \mathcal{T}$ et $B \subseteq \mathcal{I}$, on définit :

— $int(A) = \{it \in \mathcal{I} | \forall t \in A, cov(t, it)\}$

— $ext(B) = \{t \in \mathcal{T} | \forall it \in B, cov(t, it)\}$

Un ensemble d'items $C \subseteq \mathcal{I}$ est clos s'il respecte la propriété :

$$clos(C) \Leftrightarrow C = int(ext(C))$$

Autrement dit, un ensemble d'items C est un motif clos si les seuls éléments en commun des transactions qui contiennent C sont les éléments de C.

Cette restriction permet de réduire efficacement le nombre d'ensembles de termes à évaluer pour identifier les concepts dont les clusters sont déduits. Un cluster comprend tous les documents du corpus qui contiennent tous les éléments de son concept clos.

Pour exploiter les ensembles clos, les auteurs de (Ouali et al., 2016) proposent de formuler un problème dans un formalisme déclaratif en employant de la Programmation Linéaire en Nombre Entier (PLNE, voir section 3.3.2.5). Dans cette approche, tous les motifs d'éléments clos du jeu de données sont d'abord identifiés. Ensuite, les éléments correspondant à ces concepts sont groupés et ces groupements sont identifiés comme des candidats de cluster pour la partition résultat. Le modèle en PLNE formulé permet ainsi de ne retenir que les clusters optimaux étant donné une mesure de la pertinence des clusters identifiés. Il est formulé ainsi :

$$\begin{aligned} &\text{optimiser} \sum_{c \in C} v_c \cdot x_c \\ \text{tel que } &(1) \sum_{c \in C} a_{t,c} \cdot x_c = 1, \forall t \in \mathcal{T} \\ &(2) \sum_{c \in C} x_c = k_0 \\ &x_c \in \{0,1\}, c \in C \end{aligned} \tag{3.2}$$

Dans cette formulation, $\mathcal{T}$ est l'ensemble des m transactions à clusteriser, C est l'ensemble des p motifs clos qui identifient les clusters, a est une matrice binaire $m \times p$ où $a_{t,c}$ vaut 1 si la transaction t appartient au cluster associé au motif c, et v_c est une mesure de la pertinence du cluster associé au mot c. Ainsi, le modèle optimise la pertinence des clusters qu'il choisit pour faire partie de la partition, en (1) s'assurant que chaque transaction n'est couverte que par un motif et (2) en ne sélectionnant que k_0 CFTs, soit k_0 clusters, au maximum.

Des efforts ont également été menés pour limiter le plus possible le temps d'exécution des approches de clustering conceptuel appliqué à la fouille de texte. (GVR et al., 2010) propose de commencer par chercher des CFT dans le jeu de données pour identifier des clusters initiaux avec l'algorithme Apriori (Agrawal et al., 1994). Cet algorithme est fondamental en recherche de motifs d'association fréquents et est beaucoup utilisé en clustering conceptuel de texte. Il est décrit dans l'algorithme 10. Il s'appuie sur la notion de *grand* motif. Un ensemble d'attributs est un *grand* motif si son support dépasse une valeur paramétrée. Le support d'un motif est le nombre de transactions dans lesquelles il apparaît.

La fonction $apriori - gen$ consiste en deux phases, présentées dans l'algorithme 11. La première consiste à générer des motifs candidats de taille k à partir des motifs de taille $k-1$. Un nouveau candidat est généré en allongeant un motif de taille $k-1$ avec le dernier attribut d'un autre motif de taille $k-1$. La seconde phase consiste à élaguer les motifs ainsi obtenus. On retire les motifs de taille k identifiés qui contiennent un sous-ensemble de taille $k-1$ qui ne fait pas partie des ensembles de taille $k-1$ identifiés à l'étape précédente. Par exemple, en citant (Agrawal et al., 1994), si L_3 est l'ensemble $\{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{1,3,5\},\{2,3,4\}\}$. Alors l'ensemble $\{\{1,2,3,4\},\{1,3,4,5\}\}$ sera initialement identifié pour former C_4, puis sera réduit à $\{\{1,2,3,4\}\}$ car l'ensemble $\{1,4,5\}$ n'apparaît pas dans L_3.

La fonction *subset* à la ligne 5 de l'algorithme Apriori est une fonction qui recherche les motifs dans une transaction, qui peut être implantée de différentes façons. L'algorithme Apriori permet de rechercher des motifs de grande taille dans les données et il est simple de

Algorithme 10: Alogrithme Apriori

input : l'ensemble de transactions $\mathcal{T}$

$L_1 \leftarrow$ {grands motifs de taille 1}
Pour $k \leftarrow 2; L_{k-1} \neq \emptyset; k++$ **faire**
    $C_k \leftarrow apriori - gen(L_{k-1})$ `/* identification des nouveaux potentiels motifs */`
    Pour chaque *transaction* $t \in \mathcal{T}$ **faire**
        $C_t \leftarrow subset(C_k, t)$ `/*` C_t `est l'ensemble des motifs candidats de` C_k `contenus dans` t `*/`
        Pour chaque *candidat* $c \in C_t$ **faire** $c.compte++$ `/*` $c.compte$ `est le nombre d'apparitions du motif` c `dans le corpus */`
    $L_k \leftarrow \{c \in C_k : c.compte \geq support\_min\}$
retourner $\bigcup_k L_k$

Algorithme 11: $apriori - gen$

input : l'ensemble L_{k-1} des grands motifs de taille $k-1$
output: l'ensemble de motifs C_k

insérer dans C_k
    sélectionner $m.attribut_1$, $m.attribut_2$, ..., $m.attribut_{k-1}$, $m'.attribut_{k-1}$
    depuis L_{k-1} m, L_{k-1} m'
où $m.attribut_1 = m'.attribut_1$, ..., $m.attribut_{k-2} = m'.attribut_{k-2}$, $m.attribut_{k-1} < m'.attribut_{k-1}$
Pour chaque *motif* $c \in C_k$ **faire**
    Pour chaque $(k-1)$*-sous-ensemble* $s \in c$ **faire**
        Si $s \notin L_{k-1}$ **alors**
            retirer s de C_k
retourner C_k

l'implanter de sorte à ce qu'il retienne également la fréquence des motifs qu'il découvre. Il peut donc être utilisé pour découvrir des candidats de motifs, dont on peut ensuite vérifier la propriété de clôture.

Dans (GVR et al., 2010), seuls les motifs respectant la propriété de clôture sont retenus parmi les motifs fréquents identifiés avec l'algorithme Apriori. Ces motifs clos forment des clusters initiaux, ou candidats, en groupant les documents qui contiennent les mots de ces motifs. Ces clusters candidats se chevauchent beaucoup du fait que les documents peuvent être composés avec un vocabulaire similaire. Pour limiter ce problème et former un clustering correct, Kiran et al. proposent de classer les clusters auxquels un document peut appartenir en fonction de sa similarité aux autres documents du cluster. Au final, un document ne peut être assigné qu'à un maximum paramétré de max_dup clusters. Les auteurs proposent de calculer la similarité entre documents pour les affecter aux clusters en se basant sur les pages Wikipedia de leurs mots. Elle a donc le défaut de ne pas pouvoir calculer la similarité entre documents par rapport à des mots qui n'ont pas de page Wikipedia.

L'approche présentée dans (Baghel and Dhir, 2010) est également basée sur l'algorithme Apriori. Chaque document est représenté par un vecteur dont les dimensions sont des concepts de WordNet et dont les valeurs sont les sommes des fréquences des mots du concept en question. WordNet (Miller, 1995) est une ressource sous la forme d'une ontologie où les mots sont reliés entre eux s'ils sont à propos du même concept. L'algorithme Apriori est appliqué sur ces représentations à base de concepts pour identifier les motifs de concepts fréquents et en déduire les clusters initiaux. Baghel et Dhir proposent ensuite de supprimer le chevauchement des clusters initiaux en assignant chaque document au cluster pour lequel il obtient le meilleur score tel que :

$$Score(doc_j, C_i) = (\sum_x n(x) \cdot freq(x, C_i)) - (\sum_{x'} n(x') \cdot freq(x', P)) \tag{3.3}$$

où le score d'un document doc_j est calculé par rapport au cluster C_i et à la partition P à laquelle appartient ce cluster. x est un concept présent dans doc_j et dans C_i, $n(x)$ est la fréquence du concept x dans doc_j et $freq(x, C_i)$ est la fréquence du concept x dans le cluster C_i. x' est un concept présent doc_j mais absent des autres documents de C_i et $freq(x', P)$ est la fréquence du concept x' dans les documents des autres clusters de P. Ce score permet d'identifier le cluster le plus approprié pour un document.

De manière générale, nous ne mobilisons pas d'approches qui mobilisent des ressources sous la forme de connaissances externes. Les données issues de Twitter sont trop disparates pour pouvoir en tirer parti.

Finalement, des approches comme (Zheng et al., 2014) relâchent la contrainte de non-chevauchement du clustering conceptuel pour permettre d'en exploiter la méthodologie dans un cadre de clustering multiple. Dans ce dernier, les composants de la partition finale obtenue peuvent se chevaucher. Dans ces travaux, les concepts initiaux sont également calculés avec l'algorithme Apriori et leur cluster est formé des documents couverts par ces concepts. Un document peut être rattaché à plusieurs clusters plutôt que seulement au meilleur. Les documents sont représentés avec des vecteurs des scores TF-IDF de leurs mots et les clusters avec la moyenne des vecteurs de leurs documents. Les clusters ayant un vecteur de représentation très similaire sont fusionnés.

Ces approches sont représentatives des méthodologies de clustering conceptuel appliquées à la fouille de texte. Des motifs de mots sont minés avec différentes techniques,

l'algorithme Apriori revenant très souvent à cette étape, et des clusters candidats sont déduits de ces motifs. Chaque cluster est constitué des documents qui contiennent les mots du motif auquel il correspond. Ensuite, une métrique permet d'assigner un document à un seul des clusters candidats auxquels il était raccroché, ou à quelques uns d'entre eux si la contrainte de non-chevauchement est relâchée. Finalement, la partition obtenue est constituée des clusters qui restent après l'assignation des documents à leur meilleur cluster.

Le clustering conceptuel peut ainsi répondre au besoin de construire des clusters de documents, tout en proposant un concept par cluster qui peut être considéré comme sa description. Toutefois, ces concepts et les clusters qui en sont déduits sont calculés à partir de représentations binaires des documents. Ces dernières ne capturent aucune sémantique des textes représentés. Comme nous souhaitons nous appuyer sur le sens d'un tweet pour l'associer à d'autres tweets, nous ne pouvons pas employer ce type d'approche.

3.3.1.2 Clustering descriptif

Dans (Dao et al., 2018), les auteurs présentent une approche permettant de construire une partition et les descriptions des clusters qui la constituent en même temps. Cette approche repose sur une formulation du problème avec deux objectifs, poursuivis parallèlement. Étant donné la matrice X de taille $n \times f$ représentant les données, avec n le nombre d'instances à clusteriser et f la dimension de leur représentation, et une matrice X' de taille $n \times r$, avec r la taille du vocabulaire de tags décrivant les instances. X'_{ij} vaut 1 si le tag j fait partie des tags décrivant l'instance i. L'objectif du problème de clustering descriptif proposé est de construire les matrices Z, représentant le clustering, et S, représentant les descriptions des clusters, telles que :

- Z est une matrice booléenne de taille $n \times k$, avec k le nombre de clusters désiré, où Z_{ic} vaut 1 si l'instance i est rattachée au cluster c ;
- S est une matrice booléenne de taille $k \times r$, où S_{cp} vaut 1 si le tag p fait partie de la description du cluster c.

La construction de chacune de ces matrices est encadrée par un ensemble de contraintes. Pour Z, elles sont :

- (C1) chaque instance est assignée à exactement un cluster : $\forall i = 1, ..., n \ : \ \sum_{c=1}^{k} Z_{ic} = 1$
- (C2) chaque cluster doit contenir au moins une instance : $\forall c = 1, ..., k \ : \ \sum_{i=1}^{n} Z_{ic} \geq 1$
- (C3) et (C4) les affectations des instances ne doivent pas construire de clusters symétriques : $Z_{11} = 1$ et $\forall i = 2, ..., n, \ \forall c = 2, ..., k \ : \ \sum_{j=1}^{i-1} Z_{jc-1} \geq Z_{ic}$

Pour S, ces contraintes sont :

- (C5) une description doit contenir au moins un tag : $\forall c = 1, ..., k \ : \ \sum_{p=1}^{r} S_{cp} \geq 1$
- (C6) si une instance est affectée à un cluster, elle doit satisfaire une proportion paramétrée de sa description : $\forall c = 1, ..., k, \ \forall i = 1, ..., n \ : \ Z_{ic} = 1 \implies \sum_{p=1}^{r} S_{cp}(1 - X'_{ip}) \leq \alpha$. α est donc le nombre autorisé de tags n'appartenant pas à la description d'un cluster qu'une instance qui lui est rattachée peut contenir
- (C7) si un tag est affecté à la description d'un cluster, ce tag doit décrire une proportion paramétrée des instances de ce cluster : $\forall c = 1, ..., k, \ \forall p = 1, ..., r \ : \ \sum_{i=1}^{n} Z_{ic}(1 - X'_{ip}) \leq \beta$. β est donc le nombre autorisé d'instances qu'un tag ne couvrira pas dans un cluster pour pouvoir être affecté à sa description

Pour construire le clustering et les descriptions en même temps, le modèle répondant au problème doit avoir deux objectifs : un pour guider la construction du clustering et un pour construire les descriptions. Dao et al. discutent différents objectifs de clustering et de description, adaptés à différentes applications de leur méthodologie. Dans le cadre de nos travaux, nous cherchons à construire des clusters compacts et la description doit s'adapter à une répartition très éparse des tags sur les instances. En effet, d'une part la compacité des groupements de messages reflète, par construction de l'espace de représentation des documents, leur proximité sémantique. D'autre part, les tags sont ici les mots des documents, ces derniers comptant au maximum une vingtaine de mots parmi un vocabulaire d'une taille se chiffrant en milliers.

Ainsi, nous nous intéressons à l'objectif de minimisation de distance au centroïde pour le clustering et à celui de minimisation de celui appelé "désaccord de tag α-β". Notons f le premier et g le second. Ainsi f est formulé avec :

$$f = \texttt{minimiser} \sum_{i,j=1,\ i<j}^{n} Z_i Z_j^{\intercal} dist(X_i, X_j)$$

et g est de la forme :

$$g = \texttt{minimiser}\ \alpha + \beta$$

Notons que l'objectif g autorise ainsi certaines instances au sein des clusters à ne pas avoir de tags en commun.

Quel que soit le cas d'application, les deux objectifs du problème dépendent l'un de l'autre tout en étant concurrents. Ainsi, les solutions au problème, soit les affectations dans les matrices Z et S, sont réparties sur un front d'optimums de Pareto. Un front de Pareto est l'ensemble des solutions où l'amélioration d'un des objectifs entraîne nécessairement une dégradation des autres. Dans le cas présent, cela pourrait par exemple être l'affectation d'une instance à un cluster réduisant la moyenne des distances au centroïde de la partition, mais dont les tags ne correspondent pas à la description de ce cluster.

Cette approche permet de construire simultanément les clusters et leur description. Notre cas d'étude n'est toutefois pas compatible avec cette approche. En effet, elle s'appuie sur le fait que les caractéristiques des instances soient détachées de leurs tags de description, ces derniers étant des attributs binaires. Ainsi, au moment de lancer cette approche, les descripteurs sont déjà connus.

Dans notre cas, ces derniers doivent être découverts. Pour rendre nos données compatibles à cette méthodologie, on pourrait alors envisager d'utiliser des vecteurs de plongement pour représenter les instances et de les décrire avec des vecteurs de sac de mots binaires. D'une part, cela ferait porter les deux objectifs sur les mêmes caractéristiques des documents, à savoir les mots qui les composent. D'autre part, cela reviendrait à considérer tous les mots de tous les documents comme des descripteurs potentiels. Nous n'employons donc pas cette approche.

3.3.1.3 Clustering descriptif profond

(Zhang and Davidson, 2021) présentent une approche de clustering descriptif profond. Par rapport au clustering descriptif que nous avons présenté en section précédente, la principale différence réside dans le fait que les descriptions impactent plus que l'assignation

des instances aux clusters. Le modèle proposé est construit de sorte à ce que les descriptions permettent d'améliorer le modèle qui opère le clustering mais également celui qui calcule les représentations des instances, leur encodeur.

Étant donné les N instances à partitionner $x_1, ..., x_N$ où x_i est un vecteur de dimension E, le modèle f_θ qui encode les données d'entrées et K le nombre de clusters à obtenir. L'objectif du module de clustering est de calculer l'affectation y_i dans le clustering pour chaque x_i en maximisant leur information mutuelle. Pour cela, le clustering minimise la fonction de perte $\mathcal{L}_{MI}$ telle que :

$$\mathcal{L}_{MI} = \frac{1}{N}\sum_{i=1}^{N} h(p(y_i|x_i, f_\theta)) - h(\frac{1}{N}\sum_{i=1}^{N} p(y_i|x_i, f_\theta))$$

où h est la fonction d'entropie, et $p(y_i|x_i, f_\theta)$ le vecteur d'assignation à son cluster de l'instance x_i calculé avec le modèle f_θ et la fonction softmax à K sorties.

À partir de ces assignations, Zhang et Davidson proposent de calculer les descriptions des clusters à partir de l'ensemble de tags $T = t_1, ..., t_n$ décrivant les instances, où $t_i = (t_{i1}, ..., t_{iM})$ est un vecteur binaire de taille M. Les descriptions sont calculées avec l'objectif de minimiser leur taille, soit :

$$argmin_W \sum_{i,j} W_{ij}$$

où W est une matrice binaire de taille $K \times M$ telle que W_{ij} vaut 1 si le cluster i est décrit par le tag j.

En définissant la proportion du cluster i décrite par le tag j avec $Q_{ij} = \frac{1}{C_i}\sum_{t_k \in C_i} t_{kj}$, la première contrainte du module construisant les descriptions est qu'elles contiennent au moins α tags, soit :

$$\forall i \in \{1, ..., K\} : \sum_{j=1}^{M} W_{ij}Q_{ij} \geq \alpha$$

Cette contrainte amène implicitement le modèle à choisir les tags les plus couvrants pour les clusters pour satisfaire α, puisque son objectif est de construire des descriptions courtes.

L'autre contrainte du module de description vise à limiter le chevauchement entre les descriptions. Un même tag ne peut apparaître que dans β descriptions différentes, soit :

$$\forall j \in \{1, ..., M\} : \sum_{i=1}^{K} W_{ij}Q_{ij} \geq \beta$$

La solution obtenue par le modèle qui calcule les descriptions pour les clusters est notée W^*. À partir de cette solution, les auteurs définissent une fonction g telle que $g(t_i) = t_i * G$, où $G \in \mathbb{R}^{M \times M}$ est une matrice diagonale telle que G_{jj} vaut 1 si le tag j est utilisé dans W^*. Cette fonction g est alors utilisée pour identifier des paires d'instances (x_i, x_j) qui sont décrites de manière similaire mais qui n'ont pas été groupées par le clustering (soit $g(t_i) \approx g(t_j)$ et $f_\theta(x_i) \neq f_\theta(x_j)$). Pour cela, le modèle cherche, pour chaque x_i, les l instances x_j qui obtiennent les plus basses valeurs de la fonction J telle que :

$$J_i = min_{j \in \{1,...,N\}} \gamma * |g(t_i) - g(t_j)| - |f_\theta(x_i) - f_\theta(x_j)|$$

où γ est un poids qui pénalise l'écart dans l'espace de représentation en augmentant sa valeur dans le calcul de J_i.

Une fois ces l paires identifiées, la perte induite par la séparation des instances qui auraient du être assignées au même cluster est calculée avec la fonction de perte $\mathcal{L}_P$ telle que :

$$\mathcal{L}_P = \frac{1}{Nl} \sum_{i=1}^{N} \sum_{j=1}^{l} D_{KL}(p(y_i|x_i, f_\theta), p(y_j|x_j, f_\theta))$$

où D_{KL} est la divergence de Kullback-Leibler (voir section 3.4).

La fonction de perte utilisée pour calculer les mises à jour de f_θ pendant l'entraînement du modèle est finalement la fonction $\mathcal{L}$ telle que :

$$\mathcal{L} = \lambda \mathcal{L}_P + \mathcal{L}_{MI}$$

Les paramètres α, l, λ et γ du modèle sont déterminés empiriquement. Le paramètre β est déterminé itérativement. Le module qui calcule les descriptions commence à les rechercher avec $\beta = 0$. Si pour cette valeur il n'existe pas de solution W^*, la valeur de β est incrémentée de 1 et la recherche de W^* est relancée.

L'architecture proposée dans (Zhang and Davidson, 2021) permet d'entraîner un modèle de clustering et le modèle calculant les représentations sur lesquelles se base ce clustering en même temps. Les erreurs de partitionnement et de plongement des entrées sont identifiées grâce aux incohérences entre les descriptions des clusters à une étape donnée et leur constitution.

Cette approche ne correspond pas à ce dont nous avons besoin dans le cadre de nos travaux. En effet, comme pour le clustering descriptif, les instances à clusteriser sont décrites avec des attributs binaires, issus d'une connaissance externe complémentaire de leur contenu. Nous ne disposons pas de ce type d'information pour notre sujet d'étude. Les mots des documents pourraient être employés comme leurs tags pour remplacer cette connaissance externe. Cela reviendrait toutefois à nouveau à faire porter les deux objectifs du modèle, clustering et description, sur la même base. L'intérêt des approches de clustering descriptif consiste justement à mettre à profit deux sources d'informations différentes sur les instances. Notre cas d'étude n'est ainsi pas compatible avec ces approches.

3.3.2 Construire les clusters puis les décrire

Nous nous intéressons maintenant aux méthodologies où une description est construite une fois que son cluster est identifié.

3.3.2.1 BERTopic

En travaillant directement au sein de chaque cluster, on peut s'appuyer sur les statistiques caractérisant leurs éléments. C'est ce que propose de faire Grootendorst avec le modèle BERTopic proposé dans (Grootendorst, 2022). L'approche proposée se divise en trois parties.

D'abord les documents sont représentés en utilisant le modèle SBERT (Reimers and Gurevych, 2019), une adaptation du modèle BERT dont nous avons déjà parlé en section

1.3.4.6. Essentiellement, SBERT est un modèle pré-entraîné de plongement de documents et de phrases, entraîné de la même manière que BERT avec un composant de maxpooling qui calcule les représentations des phrases à partir de leurs mots.

Grootendorst propose ensuite de construire un clustering initial en utilisant l'algorithme HDBSCAN (McInnes et al., 2017). Ce dernier est une variante hiérarchique de l'algorithme de clustering par densité présenté en section 2.1.5. Nous avons vu que DBSCAN explore l'espace de représentation des instances à clusteriser de proche en proche, en agglomérant dans les mêmes clusters les points distants de moins qu'une limite ϵ. HDBSCAN fonctionne de la même manière, en mettant en place une exploration hiérarchique de différentes valeurs de ϵ. Ainsi, avec le même raisonnement, des clusters de différentes densités peuvent être identifiés.

Le clustering initial comporte un très grand nombre de clusters. Grootendorst propose de finalement réduire au nombre K de clusters désiré en fusionnant les clusters similaires. Pour identifier ces derniers, il propose une variante du TF-IDF notée c-TF-IDF, pour"*class*"-TF-IDF, qui se calcule pour un mot w en fonction d'un cluster c tel que :

$$c - TF - IDF(w, c) = tf(w, c) \cdot log(1 + \frac{A}{tf(w)}) \tag{3.4}$$

où $tf(w, c)$ est le nombre d'occurrences de w dans c, A est le nombre moyen de mots par cluster et $tf(w)$ est le nombre d'occurrences de w dans tout le corpus.

Notons que dans ces travaux, un cluster est traité comme un seul grand document en concaténant les instances. Les dix mots ayant les plus hauts scores c-TF-IDF par classe sont identifiés dans chaque cluster pour en représenter le sujet. Finalement, le cluster dont le sujet est le moins fréquent dans le corpus, c'est-à-dire celui comptant le moins de documents le représentant, est fusionné avec le cluster dont le sujet lui est le plus similaire. Cette fusion est répétée itérativement jusqu'à ce qu'il ne reste que K clusters.

Cette approche identifie des sujets cohérents et les mots qui les constituent permettent de les comprendre facilement. Toutefois, elle considère une très grande partie des données traitées comme du bruit lorsqu'elle est appliquée sur des tweets. Lors d'un test sur un de nos jeux de données, plus des trois quarts des messages ont été écartés car considérés comme du bruit. Elle ne semble donc pas adaptée à notre cas d'usage. Nous retenons toutefois la méthodologie visant à adapter les mesures statistiques des mots des documents au contexte de clustering.

Notons que nous nous intéressons à des groupements de documents construits en s'appuyant sur la sémantique capturée par les vecteurs qui les représentent. Dans les données venant de Twitter, la distinction sémantique entre les messages peut être ténue. Cela est notamment dû à deux de leurs caractéristiques. D'une part, les messages sont courts et présentent donc peu de matière pour les distinguer. D'autre part, rien ne garantit que les utilisateurs de la plate-forme ne vont pas utiliser un vocabulaire similaire pour parler de choses différentes.

Ainsi, même dans le cas de figure où l'on identifie les tags directement pour les clusters, il se peut que l'un d'entre eux soit également, sinon plus, approprié pour décrire un autre cluster. Ainsi, les mots identifiés avec la méthode choisie pour détecter les mots utiles pour la description deviennent des candidats pour décrire les clusters de documents. Comme nous cherchons à produire la meilleure liste de descriptions à l'échelle du corpus, nous nous intéressons à la manière d'assigner ces candidats aux descriptions de clusters.

Essentiellement, cela revient à exprimer un objectif pour guider la construction des descriptions et des contraintes pour qu'elles satisfassent des critères de qualité souhaités.

3.3.2.2 Problème de description de clusters

Dans (Davidson et al., 2018), Davidson et al. proposent une formulation du problème de description de cluster. Soit un ensemble d'éléments $D = d_1, ..., d_N$ et une partition $\mathcal{P}$ de D à K clusters. Étant donné un ensemble de tags T, avec un ensemble $t_i \subseteq T$ associé à chaque élément d_i, l'objectif est d'identifier un sous-ensemble $T_j \subseteq T$ pour chaque cluster C_j de sorte à satisfaire :

- chaque cluster est décrit par au moins un tag associé à un de ses éléments, soit $\forall d_i \in C_j,\ \forall j \in [1, ..., K]\ :\ |T_j \cap t_i| \geq 1$
- les ensembles $T_1, T_2, ..., T_K$ sont deux-à-deux disjoints, soit $\forall i, j \in [1, ..., K],\ i \neq j$: $T_i \cap T_j = \emptyset$
- la taille des descriptions est minimale, soit la somme $\sum_{j=1}^{K} |T_j|$ est minimisée

Cette formulation du problème est proche de notre besoin. Nous avons toutefois besoin d'adapter l'objectif du modèle ainsi que ses contraintes de taille de description et de disjonction des descriptions comme nous le verrons dans la section 5.1.3.

Cette façon de modéliser le problème correspond à une situation où les descripteurs ne sont pas issus du contenu des messages partitionnés. Davidson et al. utilisent comme tags possibles les hashtags utilisés par les auteurs des tweets du corpus d'étude. Nous souhaitons travailler directement à partir du contenu des messages. Nous avons donc besoin d'une méthodologie pour identifier quels mots sont pertinents pour être utilisés comme descripteurs.

3.3.2.3 Formalisme déclaratif

Les critères d'une bonne description peuvent être amenés à changer d'un cas d'application à l'autre. La qualité d'une description ne s'exprime par exemple pas de la même manière pour parler de sport que de politique. Le mode de formulation du problème de leur construction doit donc pouvoir s'adapter à cette problématique. Nous proposons donc d'employer un formalisme de programmation déclarative. Il permet d'exprimer ce que le modèle d'assignation doit obtenir plutôt que comment. Par exemple, exprimer que les descriptions ne doivent pas dépasser une certaine taille plutôt qu'écrire une structure de contrôle qui vérifie la taille de la description et autorise qu'on y ajoute un tag.

Le problème ainsi formulé doit avoir un objectif à maximiser, ou à minimiser, pour orienter sa résolution. Notons $f(e_\mathcal{C})$ la fonction calculant un score pour la description $e_\mathcal{C}$ du cluster $\mathcal{C}$, ce score représentant la qualité de la description. L'objectif du problème d'assignation de tag aux clusters d'une partition $\mathcal{P}$ pour former son ensemble de descriptions $E_\mathcal{P}$ prend alors la forme :

$$\texttt{maximiser} \sum_{e_\mathcal{C} \in E_\mathcal{P}} f(e_\mathcal{C}) \tag{3.5}$$

Une description $e_\mathcal{C}$ étant un ensemble de tags attribué à un cluster, on peut utiliser une fonction g calculant le score d'un mot selon le cluster auquel il serait attribué. Ainsi

on a $f(e_{\mathcal{C}}) = \sum_{w \in e_{\mathcal{C}}} g(w, \mathcal{C})$ et l'objectif devient :

$$\texttt{maximiser} \sum_{e_{\mathcal{C}} \in E_{\mathcal{P}}} \sum_{w \in e_{\mathcal{C}}} g(w, \mathcal{C}) \tag{3.6}$$

Les critères encadrant la construction des descriptions deviennent des éléments interchangeables d'une application à l'autre.

3.3.2.4 Caractéristiques d'une bonne description

La tâche de description d'ensembles de documents est très similaire au résumé ou à l'extraction de mots clefs depuis ces corpus. On peut utiliser leurs critères pour identifier les caractéristiques de "bonnes" descriptions. Dans le domaine du résumé automatique, les critères sont « traditionnellement [...] la cohérence, la concision, la grammaticalité, la lisibilité et le contenu » (Nyzam, 2021). Dans celui de la modélisation thématique, même s'il n'existe pas de méthodologie unifiée pour évaluer les modèles [1], il existe des critères récurrents. Dans (Abdelrazek et al., 2022), les auteurs identifient comme tels « perplexity, interpretability, stability, topic, diversity, efficiency, and flexibility ».

Parmi tous ces critères, nous ne nous intéressons qu'à ceux qui peuvent se mesurer sur les éléments des descriptions. C'est une condition nécessaire pour pouvoir formuler les contraintes guidant leur construction. En effet, la perplexité est une mesure de probabilité pour un modèle, permettant de vérifier que celui-ci a effectivement tiré des informations de son entraînement. La stabilité ne peut être estimée que d'une exécution du modèle à l'autre. L'efficacité est évaluée en fonction du temps de calcul du modèle pour obtenir une description. La flexibilité du modèle, le fait qu'il obtienne de bons résultats sur des données différentes de ses données d'entraînement, ne peut pas se mesurer quantitativement. Nous nous concentrons donc sur les métriques compatibles avec notre cas d'étude.

Grammaticalité C'est un critère de qualité d'un résumé estimant sa correction grammaticale. Comme nous extrayons des mots depuis des tweets, nous n'avons aucune garantie sur la correction du contenu initial. De plus, la tâche consistant à décrire ce contenu, il n'y pas d'intérêt à le corriger au moment d'en rendre compte.

Cohérence et interprétabilité La cohérence d'un résumé ou d'un sujet identifié par un modèle thématique est une mesure très répandue de leur qualité. Elle permet de mesurer si le modèle à l'origine du résumé a bien fait de placer ensemble les éléments qui le composent. Dans (Newman et al., 2010), les auteurs explorent en profondeur la tâche de l'évaluation de ce critère de manière automatique. La sémantique d'un résumé est usuellement estimée manuellement mais c'est un processus très coûteux en temps humain et donc difficile à mettre en oeuvre. Dans leurs travaux, Newman et al. montrent que la cohérence d'un résumé peut être estimée automatiquement avec presque autant de précision qu'avec une évaluation humaine. Pour cela, différentes mesures peuvent être utilisées et dans (Mehrotra et al., 2013; Abdelrazek et al., 2022), c'est la PMI (*Pointwise Mutual Information*) qui a été retenue. L'information mutuelle entre deux variables aléatoires quantifie leur dépendance et elle peut se mesurer en un point de leur distribution. Cette

1. « there is no single method for evaluating topic models » (Mehrotra et al., 2013)

mesure est l'estimateur de cohérence le plus robuste identifié dans la littérature. Dans le cas du résumé automatique, on peut la mesurer entre deux mots w et w' avec :

$$PMI(w, w') = \log \frac{p(w, w')}{p(w)p(w')} \tag{3.7}$$

où les probabilités p sont mesurées empiriquement dans le corpus d'étude.

Cette mesure permet également d'estimer l'interprétabilité. « One way of assessing the interpretability is to evaluate the coherence of the words in the generated topic » (Abdelrazek et al., 2022). En effet, un ensemble de mots cohérents est plus simple à appréhender. Notons toutefois que la cohérence ne garantit pas l'interprétabilité au sens humain d'une description. Elle peut être constituée de mots qui apparaissent effectivement ensemble dans les données, obtenant donc une bonne PMI. Mais rien ne garantit qu'un utilisateur humain pourra tirer du sens de cet ensemble cohérent. Cela ne pourrait s'évaluer que manuellement, et n'est pas compatible avec l'expression d'une contrainte.

La PMI permet d'attribuer un score à une description mais n'est pas totalement adaptée à la formulation de contrainte. En effet, on peut envisager d'utiliser la cohérence comme un objectif à maximiser, à mettre en perspective avec un autre objectif pour comparer leur intérêt. Utiliser la cohérence au sein d'une contrainte imposerait en revanche d'en fixer un seuil par description pour pouvoir formuler la contrainte.

Concision L'objectif principal de la description d'un cluster de documents est de rendre son contenu facilement accessible à la compréhension humaine. Si elle trop longue, elle sera plus difficile à comprendre pour le lecteur. Ce critère est facilement exprimable avec une contrainte, en limitant la taille maximum que peut atteindre une description. Cette contrainte serait de la forme :

$$\forall\ e_{\mathcal{C}}\ in\ E_{\mathcal{P}}, |e_{\mathcal{C}}| \leq s_{max} \tag{3.8}$$

avec s_{max} la taille maximale désirée pour une description.

Couverture de contenu Les éléments les plus fréquents dans un ensemble de documents ne sont pas forcément ceux qui portent le plus d'information. Nous l'avons vu avec certaines des mesures statistiques que nous avons présentées, comme le TF-IDF (voir section 1.2.3). Toutefois, une description doit être représentative du contenu dont elle traite. Elle doit donc couvrir une portion significative des messages du cluster. C'est l'idée que l'on retrouve derrière le modèle MCMR (*Maximum Coverage Minimum Redundancy)* de (Alguliev et al., 2011). L'objectif de ce modèle est de couvrir un maximum des phrases saillantes du ou des documents à résumer avec un minimum de redondance. L'objectif du modèle est de maximiser une mesure de similarité entre les phrases qu'il sélectionne, soit :

$$maximiser\ sim(\vec{D}, \vec{S})$$

où $\vec{D}$ est la représentation du ou des documents à résumer, $\vec{S}$ celle des phrases extraites et sim une mesure de similarité entre ces vecteurs. Nous détaillons cette approche en section 3.3.2.5. Maximiser cette similarité incitera le modèle à sélectionner plus de phrases et à ainsi couvrir une plus large portion des données à résumer.

La contrainte issue de ce critère peut être formulée de différentes façons, puisqu'il existe différentes façons d'exprimer la couverture. Nous le détaillons dans la section 5.3.1.5. Toutefois, quelle que soit la couverture utilisée pour formuler la contrainte, elle sera globalement de la forme :

$$\forall\ e_{\mathcal{C}}\ in\ E_{\mathcal{P}}, cov(e_{\mathcal{C}}, \mathcal{C}) \geq \alpha|\mathcal{C}| \tag{3.9}$$

où α est le niveau de couverture des données souhaité pour les descriptions et *cov* la mesure de couverture utilisée.

Diversité La "diversité" décrit à quel point les sujets obtenus sont sémantiquement diversifiés [2]. Plus les mots employés au sein des sujets ou descriptions sont diversifiés, meilleure est la valeur de cette métrique pour l'ensemble obtenu. Elle se calcule simplement avec :

$$Div(\mathcal{P}) = \frac{|\{\forall\ e_{\mathcal{C}}\ in\ E_{\mathcal{P}}, \forall\ w \in e_{\mathcal{C}}\ :\ unique(w, E_{\mathcal{P}})\}|}{\sum_{e_{\mathcal{C}}\ in\ E_{\mathcal{P}}} |e_{\mathcal{C}}|} \tag{3.10}$$

où $unique(w, E_{\mathcal{P}})$ signifie que le mot w n'est utilisé que dans une seule description de $E_{\mathcal{P}}$ et aucune autre.

Cette diversité a différents noms dans la littérature, où l'on parle également de non redondance ou de non chevauchement entre les descriptions. On peut la considérer comme un critère de qualité car un ensemble de descriptions hautement redondant sera mécaniquement moins représentatif du contenu à décrire. Toutefois, forcer l'unicité d'un mot au sein d'un ensemble de descriptions est un critère très fort qui peut limiter la faisabilité du problème d'assignation de tag. Aussi, on peut formuler la contrainte de manière à autoriser un certain niveau de chevauchement en lui donnant la forme :

$$\forall\ e_{\mathcal{C}}\ in\ E_{\mathcal{P}}, \forall\ w\ in\ e_{\mathcal{C}},\ Occ(w, E_{\mathcal{P}}) \leq \beta \tag{3.11}$$

où $Occ(w, E_{\mathcal{P}})$ est le nombre de descriptions différentes de $E_{\mathcal{P}}$ où w apparaît et β la redondance maximum souhaitée pour un mot.

Contraintes et cas d'usage Formuler des contraintes qui guident la construction des descriptions doit être fait en considérant la nature des données traitées. En effet, la diversité d'un ensemble de descriptions est par exemple inféodé à la partition de documents qui est décrite. Dans des données en langue naturelle, et particulièrement dans celles venant de Twitter, il n'y a aucune garantie que les vocabulaires employés pour parler de deux sujets différents soient clairement distincts. Ainsi, comme les représentations des documents reflètent cette distinction, il n'y aucune garantie de pouvoir identifier des groupes de documents clairement séparés. En conséquence, la diversité de leur description serait mauvaise sans que cela signifie que celles-ci soient mauvaises. De plus, il est naturel que les descriptions de deux ensembles de documents à propos de sujets similaires soient elles-mêmes similaires. Il faut donc adapter les contraintes et les éventuels seuils qu'elles fixent au cas traité et au besoin formulé par l'utilisateur qui devra exploiter les descriptions.

2. « This metric describes how semantically diverse the obtained topics are » (Abdelrazek et al., 2022)

3.3.2.5 Formulation du modèle d'assignation

Le problème d'assignation de tag aux descriptions est donc un problème d'optimisation comprenant des contraintes. Pour le résoudre, on peut naturellement s'orienter vers la Programmation Par Contrainte (PPC). La PPC est une méthode de résolution de problèmes combinatoires qui repose sur la spécification de contraintes logiques entre les variables. Elle consiste à définir un ensemble de variables avec leurs domaines de valeurs, ainsi que des contraintes qui restreignent les combinaisons de valeurs possibles pour ces variables. Dans le cadre de nos travaux, les unités à assigner, les tags de descriptions, sont atomiques : un tag est assigné ou n'est pas assigné à une description. Ainsi nous devons formuler nos contraintes avec des valeurs entières. La PPC n'est donc pas l'approche la plus appropriée pour notre problème. Nous proposons donc d'employer la PLNE.

La PLNE est une branche de l'optimisation dans laquelle les problèmes sont exprimés avec un objectif et des contraintes linéaires. Ils sont plus complexes que les problèmes d'optimisations linéaires classiques car les variables doivent prendre leurs valeurs dans $\mathbb{Z}$. Notons $x \in \mathbb{Z}^n$ le vecteur de variables entières à décider par le modèle, $c \in \mathbb{R}^n$ le vecteur représentant les valeurs attribuées à ces variables, $A \in \mathbb{R}^{m \times n}$ une matrice représentant les attributs des objets traités par le modèle et $b \in \mathbb{R}^n$ une contrainte sur la valeur de ces attributs. La forme canonique d'un modèle de PLNE est :

$$\begin{aligned} &\texttt{maximiser } c^\intercal x \\ &\texttt{tel que } Ax \leq b \end{aligned} \tag{3.12}$$

Dans le cadre de nos travaux, nous cherchons à obtenir les meilleures descriptions possibles à partir d'un ensemble de descripteurs. Ainsi, la forme générique de l'objectif du modèle que nous utiliserions serait :

$$\texttt{maximiser} \sum_{e_\mathcal{C} \in E_\mathcal{P}} f(e_\mathcal{C}) \tag{3.13}$$

où f est la fonction qui évalue la qualité de la description $e_\mathcal{C}$.

Dans la littérature, il existe d'autres approches mobilisant la PLNE pour construire des résumés d'ensembles de documents. Toutefois, à notre connaissance, le problème est toujours fondamentalement le même. Il consiste à établir un score représentant la pertinence des mots ou des phrases candidats et à les sélectionner en maximisant ce score tout en respectant les contraintes identifiées par les auteurs. Nous avons déjà mentionné (Alguliev et al., 2011) et le modèle *MCMR* proposé dans ces travaux, qui représente cette façon de formuler le problème. Nous avons présenté une écriture simplifiée du modèle *MCMR*. Même en donnant tout son détail, il reste simple dans sa formulation :

$$\begin{aligned} &\texttt{maximiser } f_\alpha = \alpha f_{cos} + (1-\alpha) f_{NGD} \\ &\texttt{tel que } \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (len(s_i) + len(s_j)) X_{ij} \leq L \end{aligned} \tag{3.14}$$

où n est le nombre de phrases candidates pour décrire l'ensemble de documents D, X_{ij} est la variable décidant si la paire de phrases (s_i, s_j) est sélectionnée pour décrire D et L la longueur maximale de la description de D.

f_{cos} et f_{NGD} sont telles que :

$$f_{cos} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (sim_{cos}(\vec{D}, \vec{s_i}) + sim_{cos}(\vec{D}, \vec{s_j}) - sim_{cos}(\vec{s_i}, \vec{s_j}))X_{ij} \tag{3.15}$$

et :

$$f_{NGD} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (sim_{NGD}(\vec{D}, \vec{s_i}) + sim_{NGD}(\vec{D}, \vec{s_j}) - sim_{NGD}(\vec{s_i}, \vec{s_j}))X_{ij} \tag{3.16}$$

où $\vec{s_i}$ est la représentation vectorielle de la phrase s_i. Le paramètre α dans l'équation 3.14 permet d'ajuster l'importance donnée à la similarité du cosinus et à celle de la *Natural Google Distance* en fonction du cas d'application et des données traitées. La similarité de la *NGD* se calcule comme suit :

$$sim_{NGD}(s_i, s_j) = \frac{\sum_{w_k \in s_i} \sum_{w_l \in s_j} sim_{NGD}(w_k, w_l)}{|s_i| \cdot |s_j|}$$

avec :

$$sim_{NGD}(w_k, w_l) = \frac{\max(\log(f_{w_k}), \log(f_{w_l})) - \log(f_{w_k w_l})}{\log n - \min(\log(f_{w_k}), \log(f_{w_l}))}$$

où f_{w_k} est le nombre de phrases du corpus contenant le mot w_k et $f_{w_k w_l}$ est le nombre de phrases contenant les deux mots w_k et w_l.

De fait, la sim_{NGD} est une mesure du volume de co-occurrence des mots pour en calculer la similarité, là où la similarité du cosinus ne prend en compte que le fait que les mots co-occurrent. Ainsi le paramètre α dans l'équation 3.14 permet d'équilibrer entre une similarité prenant en compte la magnitude des vecteurs (sim_{NGD}) et une s'appuyant seulement sur les angles qu'ils forment.

Ce modèle est fondamentalement simple. Il maximise une mesure de similarité au sein de la description, ce qui l'amène à sélectionner les phrases qui ressemblent le plus aux autres phrases du corpus. Comme le score doit être maximisé et qu'il est calculé par somme, le modèle sélectionnera autant de phrases que le paramètre L lui permet.

Ce mode de formulation permet de mettre à profit la PLNE pour construire un modèle d'assignation de tags aux descriptions de cluster dans un formalisme déclaratif.

3.4 Évaluer les descriptions du modèle

Une fois les descriptions obtenues, il faut pouvoir les évaluer pour voir si elles sont de bonne qualité. Les critères utilisés pour formuler les contraintes permettent de forcer certaines des caractéristiques des descriptions que l'on va obtenir. Cela ne peut toutefois pas suffire à évaluer la qualité globale de l'ensemble obtenu. Il faut faire appel à d'autres métriques.

Les modèles thématiques peuvent s'évaluer en employant les métriques que nous avons énoncées en parlant des contraintes. Ils peuvent également être explorés entièrement du point de vue des lois de probabilités qu'ils apprennent. Dans (Wallach et al., 2009), les

auteurs analysent différents estimateurs de probabilités qui permettent d'évaluer si les lois apprises par un modèle sont correctes. Comme nous ne mobilisons pas ce type de modèle, nous ne pouvons pas recourir à ce type d'évaluation.

Trois grands axes sont identifiés dans (Nyzam, 2021) pour l'évaluation de résumés : l'évaluation manuelle, l'évaluation automatique avec résumés humains et l'évaluation entièrement automatique.

L'évaluation manuelle consiste à faire appel à des relecteurs humains pour estimer la qualité des résumés obtenus. Cela passe donc par la mise en place d'un protocole de validation sur un grand ensemble de données en nécessitant une lourde charge de travail humain. Évaluer « la qualité linguistique et la couverture du contenu [des résumés] comme dans la campagne d'évaluation DUC (Document Understanding Conference) nécessite plus de 3 000 heures d'efforts humains » (Nyzam, 2021). Une évaluation aussi coûteuse n'étant pas envisageable dans le cadre de ces travaux de thèse, nous ne présentons pas ce type de méthode.

L'évaluation s'appuyant sur une production humaine permet de contourner une partie du problème de l'évaluation manuelle. Au lieu d'évaluer tous les résultats obtenus, on construit une référence pour un jeu de données connu. On connaît donc le ou les résumés à obtenir, ces derniers ayant été écrits à la main. C'est sur cette base de raisonnement qu'a par exemple été mis au point l'outil ROUGE (pour *Recall-Oriented Understudy for Gisting Evaluation*) et son ensemble de métriques (Lin and Och, 2004). Essentiellement, cet outil permet d'estimer à quel point un résumé obtenu automatiquement a des éléments en commun avec la référence établie. Parmi les métriques de l'outil, on a par exemple ROUGE-N qui se calcule avec :

$$ROUGE - N = \frac{\sum_{S \in ResumesReference} \sum_{N-gramme \in S} Count_{match}(N - gramme)}{\sum_{S \in ResumesReference} \sum_{N-gramme \in S} Count(N - gramme)}$$

où N est la taille des n-grammes utilisée pour l'évaluation, $Count_{match}(N - gramme)$ est le nombre de fois où le n-gramme $N - gramme$ apparaît à la fois dans le résumé proposé et la référence, et $Count(N - gramme)$ est le nombre total d'apparitions de $N - gramme$ dans le résumé proposé.

Toutefois, pour pouvoir mettre en oeuvre ce type d'évaluation, il faut disposer d'une référence à laquelle se comparer. Comme cela n'est pas le cas dans le cadre de nos travaux, nous ne pouvons pas mobiliser cette méthodologie d'évaluation des descriptions.

Dans (Nyzam, 2021), l'auteur expose que la divergence de Jensen-Shanon est la meilleure mesure pour évaluer entièrement un résumé obtenu automatiquement. C'est celle qui obtient la meilleure corrélation avec l'évaluation humaine, ce qui est également établi dans (Lin et al., 2006). Cette métrique se mesure entre deux distributions. Elle s'appuie sur la divergence de Kullback-Leibler, calculée comme suit :

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{3.17}$$

La divergence de Jensen-Shannon est une version symétrique et lissée de celle de Kullback-

Leibler et se calcule avec :

$$\begin{aligned} JSD(P||Q) =& \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(M||Q) \\ & \text{avec } M = \frac{1}{2}(P+Q) \end{aligned} \tag{3.18}$$

Néanmoins Nyzam explique aussi que cette divergence de Jensen-Shannon, la meilleure parmi les mesures automatiques, ne permet pas d'obtenir des informations précises sur la qualité d'un résumé automatique en particulier. Elle permet de fournir des moyennes à l'échelle de plusieurs corpus, mais ne permet pas d'obtenir d'information sur la qualité linguistique et la sémantique d'un résumé donné.

Dans (Gao et al., 2020), les auteurs proposent d'exprimer la qualité d'un résumé automatique en fonction du nombre de phrases "saillantes" issues du texte d'origine qu'il contient. Il existe plusieurs manières d'identifier celles-ci. Les auteurs proposent d'employer une méthodologie basée sur une représentation des documents à résumer par un graphe G. Dans G, les phrases sont les sommets et les arcs sont pondérés par un score de similarité entre les phrases de ces sommets. Ce graphe permet d'identifier les phrases les plus similaires aux autres dans les données à résumer. Elles sont extraites pour en former une pseudo-référence. Celle-ci est à son tour plongée dans un espace de représentation, avec le même modèle qui a calculé les représentations des données à résumer. Finalement le score proposé, *SUPERT*, évalue la qualité du résumé en comparant la similarité entre le plongement des données à résumer et celui de leur pseudo-référence.

Dans le cadre de nos travaux, où une phrase couvre souvent l'intégralité d'un tweet, nous souhaitons travailler en utilisant les mots comme descripteurs. Ainsi, cette métrique n'est pas adaptée pour l'approche que nous voulons mettre en place.

Évaluer quantitativement et qualitativement de manière automatique des résumés pour lesquels on ne dispose pas de références construites par des humains est une tâche ouverte en recherche. À notre connaissance de la littérature actuelle, « les métriques entièrement automatiques ne conviennent pas à l'évaluation de la qualité linguistique » d'un résumé ou d'une description (Nyzam, 2021). Ainsi, estimer la qualité d'une description sans référence annotée nécessite au moins une intervention humaine partielle ou la proposition de nouvelles métriques.

3.5 Décrire des données textuelles reçues en continu

Les travaux traitant et décrivant des données textuelles reçues en continu depuis un réseau social portés à notre connaissance peuvent être regroupés sous deux tâches : la détection de différents types d'évènements et le suivi de tendances. Certaines de ces approches peuvent également répondre à ces deux besoins. La nature des descriptions construites change ainsi en fonction de l'application traitée.

3.5.1 Détection et description d'évènement

Toutes les approches qui ont retenu notre attention exploitant un réseau social, et en particulier Twitter, recherchent des changements soudains dans le comportement des utilisateurs. Comme nous nous intéressons à la description de données textuelles, nous

nous limitons aux approches de détection d'évènement travaillant à partir du contenu des messages. Nous ne parlerons donc pas ici de celles qui s'appuient sur les communautés d'utilisateurs de la plate-forme ou sur ses fonctionnalités comme le "j'aime" ou le "retweet".

Un des axes dans la détection d'évènement repose sur la représentation vectorielle des messages entrants et le clustering incrémental. Les croissances brusques des clusters mettent en évidence les évènements, qui sont décrits avec les tweets des clusters ou des mots importants parmi eux. Nous avons déjà présenté ces approches dans la section 2.3.3.

D'autres approches mettent à profit d'autres types de méthodologies. Dans (Weng et al., 2011), les auteurs utilisent le TF-IDF pour traduire les tweets en signaux. La méthodologie employée ici est la transformation en ondelettes. Nous ne détaillons pas ce procédé mathématique, proche des transformées de Fourrier, sortant du cadre de nos travaux. Essentiellement, elle consiste dans ce cas à décomposer un signal continu en un ensemble discret de coefficients. Dans ces travaux, chaque mot w est considéré comme un signal. Pour le composer, le score TF-IDF de w est calculé à intervalles réguliers. Ainsi, ce score à un instant t est considéré comme un coefficient d'ondelette. Une fenêtre temporelle glissante permet de lisser le signal formé par les scores du mot. Les tweets sont représentés en composant les signaux de leurs mots, ce qui permet d'utiliser des calculs de similarité de signal entre "signaux de tweet". Ces derniers permettent d'identifier des groupes de messages proches, pour ne pas avoir à traiter individuellement tous les messages. Les brusques changements de signaux de mots indiquent un évènement. Finalement, une description est formée du ou des tweets qui ont agrégé par le calcul de similarité de signal le plus d'autres tweets autour d'eux et qui comptent le plus de mots associés à des signaux en pic. Nous retrouvons ici une approche au fond similaire au clustering incrémental, à ceci près que la représentation utilisée s'appuie sur des méthodes de traitement de signal plutôt que de plongement lexical.

Le travail présenté dans (Li et al., 2012) segmente les tweets en des unités plus longues que des tokens et utilise de la connaissance externe pour les grouper. La première étape consiste donc à identifier ces segments. Elle consiste en un problème d'optimisation qui cherche à maximiser un score de cohésion des segments d'un tweet. Soit d un tweet d'un ensemble $\mathcal{D}$, d est considéré comme une séquence de m segments qui ne se chevauchent pas, soit $d = < s_1 s_2 .. s_m >$. En notant ce score de cohésion c, l'objectif du modèle de segmentation est

$$arg\ max_{s_1,s_2,...,s_m}\ c(d) = \sum_{i=1}^{m} c(s_i)$$

avec

$$c(s) = \mathcal{L}(s) \times e^{Q(s)} \times \sigma(SCP(s))$$

$Q(s)$ et $SCP(s)$ sont ici des scores issus de ressources externes. Pour le segment s, le score SCP dénote sa probabilité conditionnelle symétrique telle que calculé par le service Microsoft Web N-GRam [3]. Ce service renvoie ici la probabilité que les mots du segment forme un n-gramme dans ce que "connaît" ce service, à savoir un très grand ensemble de données. Cette probabilité est donc la probabilité des mots du segment d'apparaître ensemble. Le score $Q(s)$ quant à lui représente la probabilité que les mots du segment s forment une ancre, un lien vers une autre page, dans les pages Wikipédia qui contiennent s.

3. `http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx`

$\mathcal{L}(s)$ est un facteur visant à réduire le score de cohésion des segments s'ils sont plus longs, pour ne pas naturellement les favoriser. Cette segmentation s'apparente à une recherche d'entités nommées basée sur une connaissance plus large de la langue, tirée d'un service externe et de Wikipédia.

Les segments ainsi identifiés sont ensuite clusterisés selon un score de similarité qui est élevé si les segments comparés ont des TF-IDF similaires dans plusieurs des fenêtres temporelles considérées. Les segments sont filtrés pour favoriser ceux qui forment ou contiennent des ancres Wikipédia. Ceux-ci sont en effet plus faciles à comprendre et les segments plus difficiles à interpréter sans leur contexte sont ainsi éliminés. Finalement les descriptions sont formées des segments qui ont les plus fortes similarités avec le reste de leur groupe.

Dans le cadre de nos travaux, nous souhaitons nous reposer uniquement sur les données que nous collectons, sans avoir recours à une connaissance externe. Le but est d'être le plus proche possible de la réalité exprimée à travers les messages captés, sans a priori, linguistique ou sémantique, sur ce qu'ils contiennent. De plus, nous souhaitons également nous appuyer sur la sémantique que l'on peut identifier dans ces données pour les traiter. Nous retenons donc les méthodologies par clustering incrémental basé sur des plongements de mots et de documents, comme nous le verrons dans le chapitre 7.

3.5.2 Modèle thématique dynamique

Les modèles thématiques permettent, comme nous l'avons vu, d'identifier des sujets dans des données en calculant les distributions de leurs mots, pour repérer ceux qui ont plus de chances d'apparaître ensemble. Ceux que nous avons présentés en section 3.2.3 ne sont pas adaptés aux flux de données. Toutefois, modifier ces approches pour traiter ces derniers permettrait de faire le suivi des thèmes que les modèles identifient et ainsi de faire du suivi de tendances dans des données textuelles.

Une façon de répondre à ce problème est la modélisation thématique dynamique, comme proposée par (Blei and Lafferty, 2006). Elle consiste simplement à discrétiser les données reçues en continu en des ensembles finis et à y appliquer des techniques classiques comme la LDA. Les travaux de (Li, 2021) proposent d'appliquer cette méthodologie en constituant des ensembles figés dont les fenêtres temporelles se chevauchent pour pouvoir mettre en oeuvre un suivi continu des sujets.

Un autre axe, poursuivi dans les travaux de (Wang and McCallum, 2006), est de modifier directement les modèles pour les adapter aux données en flux. C'est le cas des approches de modélisation de sujet au cours du temps. Ce type de processus est génératif, comme de la modélisation de sujet classique. La modification est relativement simple à appréhender. Le modèle génère toujours une distribution de sujets θ_d spécifique à chaque document d du corpus, avec un paramètre de Dirichlet α. Étant donné un des sujets z à identifier, le modèle génère toujours une distribution des mots spécifique à chaque sujet ϕ_z, avec un paramètre de Dirichlet β. En plus de ces distributions, le modèle tient compte du temps avec une nouvelle distribution de loi bêta des datages des mots des documents notée ψ.

La distribution des sujets sur un mot dépend de celle des sujets sur le document de ce mot. L'ajustement de la loi bêta d'où sont tirés les datages des documents est opéré en considérant le datage réel des documents. Le tirage du datage étant spécifique à un

document, il est directement associé au tirage de sujet pour ce document. L'ajustement de la distribution des datages permet ainsi de faire prendre la date du document en compte par le tirage de ses sujets.

Les lois de distribution des modèles de sujets temporels sont inférées avec les mêmes outils statistiques que les modèles thématiques classiques.

Ces modèles permettent ainsi de décrire les données reçues sur un flux avec les sujets qui y sont identifiés et permettent de suivre l'évolution de ces sujets au cours du temps. Toutefois, comme indiqué par (Li, 2021), le coût computationnel de l'inférence de leurs lois augmente rapidement avec la durée d'écoute du flux. Naturellement, plus elle est longue, plus le nombre de datages de document à prendre en compte pour inférer la distribution des sujets dans le temps augmente. Ces approches sont donc coûteuses à mettre en oeuvre et ne correspondent donc à nos besoins.

Deuxième partie

Contributions

Chapitre 4

Acquisition et préparation des données

Sommaire

Nos travaux abordent le problème de description d'évènements à partir d'un flux de messages venant de Twitter. Toutefois, aussi bien dans un contexte incrémental que dans un contexte statique, où l'ensemble des données est fixé en amont, l'approche que nous proposons ne dépend pas de la nature des données traitées. En effet, les seuls composants de notre système dont l'usage est restreint aux tweets concernent les pré-traitements utilisés, ce qui rend notre approche portable à différents cas d'usage. Le modèle de plongement utilisé est également choisi de sorte à être particulièrement adapté à l'apprentissage sur des données issues de Twitter, mais il peut être appliqué sur n'importe quelles données textuelles. Ainsi, on peut considérer la partie de nos systèmes spécifique à la préparation des données indépendamment du reste de ceux-ci, et nous la présentons dans ce chapitre.

Notons qu'en adaptant cette partie de pré-traitement à d'autres données et/ou cas d'application, le reste de nos approches fonctionnerait de la même manière.

4.1 Requêtage et pré-traitement

Quelle que soit l'approche, la première étape consiste naturellement à collecter les données à traiter. Le premier point à aborder est celui des évènements que nous allons cibler.

Évènements cibles Le cas d'usage initial qui a motivé le sujet étant l'écoute du contenu de Twitter pour des aéroports, nous nous intéressons à des évènements qui pourraient impacter ce type d'activité. De plus, l'objectif est de traiter des flux dans lesquels il peut apparaître de nouveaux mots et usages de la langue. Nous nous intéressons donc à des évènements récents, plus susceptibles de présenter ce type d'évolution, pour éprouver nos hypothèses face à cette caractéristique des données.

Au moment où nous avons mené ces expérimentations, deux évènements majeurs ont retenu notre attention. Le premier a été la tempête Alex, qui a frappé la France du 30 septembre au 3 octobre 2020. Les évènements climatiques sont des cibles toutes choisies pour nos travaux, car pouvant mettre à l'arrêt toutes les activités humaines quand ils sont de forte magnitude. Il est par ailleurs utile de disposer d'un jeu de données en français, ces derniers étant moins nombreux et plus difficiles à obtenir que ceux en anglais, un des objectifs poursuivis étant l'indépendance de l'approche face à la langue traitée. L'objectif de l'approche étant d'être le plus agnostique de la langue que cela soit possible, nous souhaitons disposer de corpus d'étude dans au moins une autre langue que l'anglais. En outre, la tempête Alex était alors l'évènement climatique majeur le plus récent. Les pré-traitements appliqués doivent être les plus génériques et accessibles possible.

L'autre cible que nous avons retenue est la guerre Russo-Ukrainienne, qui a commencé en février 2022 et qui est en cours au moment de la rédaction de ce manuscrit. Les conflits de grande envergure ont des impacts à tous les niveaux partout dans le monde. Vérifier nos hypothèses sur ce type d'évènements est donc important.

4.1.1 Requêtes

La requête que nous avons utilisée pour collecter les tweets sur la tempête Alex est la suivante.

```
'query': '(tempete OR tempête OR tempet OR #tempête OR #tempet OR #tempete
    OR #alex OR alex) -is:retweet lang:fr',
'end_time':'2020-10-10T23:59:59Z',
'start_time':'2020-09-27T00:00:00Z',
'tweet.fields': 'id, text, author_id, created_at'
```

L'API de requêtage de Twitter fonctionne avec des mots-clefs et des opérateurs logiques sur les attributs de messages qu'elle indexe. Ainsi, cette requête retournera tous les tweets publiés entre le 27 septembre et le 10 octobre 2020, qui contiennent au moins un des mots-clefs entre parenthèses dans le champ `'query'`, qui ne sont pas des retweets et qui sont écrits en français. Chaque tweet est retourné sous la forme d'un objet JSON dont les champs seront l'identifiant unique du tweet dans l'API, son contenu, l'identifiant de son auteur et son horodatage. Concrètement, cette requête a retourné 20016 tweets.

Comme ces requêtes vont déterminer les données à partir desquelles nous allons travailler, il faut leur prêter une attention particulière. Notre objectif est de décrire l'évènement à partir de ses manifestations qui sont visibles sur Twitter. La requête doit donc être la moins spécifique possible, pour ne pas introduire de biais venant d'a priori sur le type d'évènement ciblé, ici une tempête, ou sur l'évènement en lui-même, ici la tempête Alex. Ainsi, nous n'employons comme mots-clefs que `'tempête'` et `'alex'`. Notons que le moteur de recherche de l'API n'est pas sensible à la casse des mots-clefs. Nous ajoutons des variations de l'orthographe de ces deux mots-clefs, pour essayer de récupérer le contenu où l'un d'eux aurait été mal saisi par l'auteur du tweet. Par ailleurs, nous incluons également les variantes en hashtags de ces mots-clefs. En effet, l'API recherche les mots-clefs de manière tokenisée. Ainsi, `'tempête'` ne permettra pas de récupérer un message contenant le hashtag `'#tempête'`.

Nous excluons par ailleurs les retweets en incluant l'opérateur `'is'`, inversé avec le `'-'` et avec la valeur `'retweet'`. Les retweets sont des copies de tweets originaux, majoritairement utilisés par les 'twittos' pour diffuser une information dans différentes communautés. Comme nous l'avons vu dans la partie I, le décompte de retweet d'un tweet peut être utilisé pour mesurer son importance comme dans (Zhao et al., 2011a), ou pour le catégoriser comme dans (Becker et al., 2011). Toutefois, nous ne nous intéressons pas à la dynamique de diffusion d'une information pour décider s'il faut ou non la prendre en compte. Notre objectif est de décrire le contenu résultant de la requête, et il n'existe aucune garantie qu'une information peu diffusée ne soit pas pertinente pour décrire l'évènement. Nous nous intéressons donc à toute information unique du corpus. Par ailleurs, si une information est fortement retweetée, elle sera diffusée par un grand nombre d'utilisateurs qui n'utilisera pas cette fonctionnalité. Ainsi le retweet n'apporte pas d'information supplémentaire dans le cadre de nos travaux.

Les dates de début et de fin que nous avons utilisées couvrent une plus large fenêtre temporelle que les dates officielles de l'évènement. Nous faisons ce choix pour tester s'il est possible de détecter des signes avant-coureurs ou des alertes précoces pour l'évènement cible avec l'architecture que nous proposons. La partie postérieure à l'évènement est ciblée pour explorer la manière dont le contenu qui s'apparente à des constats sur la cible est traitée par notre approche.

Dans les premières itérations de ces expérimentations, nous n'avions pas inclus l'opérateur `'lang'` avec la valeur `'fr'`. En explorant les résultats obtenus avec ces données, nous avons pu identifier un cluster à propos du groupe de K-Pop "DRIPPIN". Il se trouve qu'au moment où la tempête Alex frappait la France, un membre de ce groupe de musique, également appelé Alex, était mis en avant par les campagnes publicitaires promouvant la sortie de leur nouvel album. Même s'il était intéressant de constater que ce "bruit" au regard du type d'évènement cible était regroupé dans le même cluster, nous avons fait le choix de nous restreindre aux tweets en français pour l'exclure des données. La principale raison motivant cette décision était de pouvoir évaluer le degré de finesse atteint dans l'identification des manifestations de la cible. Cette granularité serait à estimer en considérant qu'elle serait issue d'un cas favorable, où un maximum de bruit a été retiré des données. Cela permettrait d'évaluer la plus fine granularité qu'il soit possible d'obtenir avec cette architecture.

La requête employée pour collecter le jeu de données sur le début de la guerre Russio-Ukrainienne est la suivante.

```
'query': '(war OR wa OR warr OR waa OR #war OR #wa OR #warr OR
    #waa OR ukraine OR ukrain OR ukrainn OR ukraiin OR ukrainee OR #ukraine
    OR #ukrain OR #ukrainn OR #ukraiin OR #ukrainee) -is:retweet lang:en',
'end_time':f'2022-03-04T23:59:59Z',
'start_time':f'2022-02-23T00:00:01Z',
'tweet.fields': 'id, text, author_id,created_at'
```

Elle est construite avec les mêmes considérations que celle pour la tempête Alex, à ceci près qu'elle concerne des données en anglais. Notons que nous avons élargi ici le nombre de variantes d'orthographe des mots-clefs `'war'` et `'ukraine'` pour estimer à quel point cela impacte le contenu récupéré. Il est apparu que les fautes de typographie ne se sont comptées qu'en dizaines dans ces deux corpus. En revanche, la présence de `'wa'` parmi

les mots-clefs a par exemple fait retourner à la requête un nombre relativement important de tweets (159) contenant de l'anglais et le mot chinois ou japonais `'wa'`. Il est ainsi apparu que d'intégrer trop de variantes orthographique des mots-clefs des cibles pouvait être contre-productif en amenant plus de bruit que de contenu pertinent. Nous avons donc reformulé la requête comme suit, et obtenu un corpus de 18311 tweets issus de la période s'étalant du 23 février au 4 mars 2022.

```
'query': '(war OR warr OR OR #war OR ukraine OR ukrain OR ukrainee
     OR #ukraine OR #ukrain OR #ukrainee) -is:retweet lang:en',
'end_time':f'2022-03-04T23:59:59Z',
'start_time':f'2022-02-23T00:00:01Z',
'tweet.fields': 'id, text, author_id,created_at'
```

4.1.2 Pré-traitements des données

On appelle "pré-traitements" les procédés que l'on applique à des données dans le but de faciliter leurs traitements, ceux-ci répondant à une tâche spécifique. Ils permettent également dans la très grande majorité des cas d'améliorer les résultats obtenus pour cette tâche.

Dans le cas présent, ces pré-traitements ont pour but d'améliorer la capture de la sémantique des mots et des documents par le modèle de plongement que nous employons. Ainsi, leur objectif est de modifier les données pour rendre la sémantique distributionnelle (voir 1.3.1) plus accessible au modèle Doc2Vec. Les vecteurs de représentations seront ainsi plus représentatifs du sens des documents qu'ils représentent. Finalement, le cluster obtenu sera plus représentatif de la distinction entre les différents sujets des données. Voilà pourquoi il faut prêter une attention toute particulière à ces pré-traitements.

C'est la seule brique de l'architecture que nous proposons qui soit inféodée à la langue du corpus traité et qui doive mobiliser des ressources externes. L'anglais et le français sont des langues similaires en terme de construction. Les mêmes types de pré-traitements peuvent s'appliquer, en adaptant les lexiques et les modèles mis en oeuvre. Les outils que nous employons pour préparer nos données sont des standards dans le domaine du TAL, de sorte à facilement pouvoir les changer en cas d'application sur une autre langue.

Suppression des mots-vides Nous retirons les mots vides, ou "*stop words*", pour deux raisons. D'abord parce qu'ils ne portent que très peu de sens au sein de leur phrase comparativement aux mots qui ne sont pas des stop words. On y retrouve des mots, comme les déterminants, qui sont nécessaires pour qu'une phrase soit correcte dans la langue dans laquelle elle est écrite mais qui n'apporte pas d'information sur son sens. Dans "le chat est dans le jardin", les "le" permettent de construire une phrase de français correcte sans apporter plus d'information sur le fait que ce chat soit dans ce jardin. La deuxième raison motivant la suppression des mots vides est le fait qu'ils puissent perturber la capture de sens du modèle de plongement. La seule sémantique que puisse capturer ce dernier est celle qui s'exprime à travers les co-occurrences des mots du corpus, qu'il "voit" à travers sa fenêtre d'apprentissage. Ainsi, si la présence d'un mot vide empêche deux mots porteurs de sens d'apparaître dans le même voisinage, le modèle ne pourra pas voir cette co-occurrence.

Exemple 4.1.1. "les vents très violents de la tempête ont déracinés des arbres"

Si cette phrase est présentée à un modèle dont la fenêtre est de taille quatre, le voisinage dans lequel le mot "violents" sera central sera :

"vents", "très", "de" et "la".

Le modèle ne pourra voir la co-occurrence entre "vents", "violents" et "déracinés". La même phrase de laquelle on a retiré les mots vides devient :

"vents très violents déracinés arbres",

et on constate que le modèle de plongement pourra dans ce cas voir la co-occurrence.

Pour les deux corpus, nous utilisons la ressource *NLTK*[1], qui prend la forme d'un dictionnaire pour une langue donnée, pour reconnaître et supprimer ces mots-vides. Notons que nous considérons également les mots-clefs de la requête envoyée à l'API comme des mots vides. Comme ils sont peu nombreux, les tweets du corpus sont très susceptibles de contenir un voire tous ces mots. Dans le cadre précis et restreint des données récupérées par la requête, les mots-clefs de cette dernière ne sont pas porteurs d'une information qu'il soit possible de distinguer du reste.

Réduction du nombre de formes d'un mot Dans les deux langues de nos corpus, les mots peuvent prendre différentes formes pour différentes raisons. Cela peut par exemple être du à la conjugaison des verbes ou à l'accord des adjectifs. Ces variations de morphologies ne sont toutefois pas utiles du point de vue de la sémantique distributionnelle. Considérons l'exemple de "vents violents" qui "déracinent des arbres". Les accords en nombre n'apportent pas d'information supplémentaire à l'association de termes que peut détecter le modèle. L'information à retenir est l'association de "vent violent" avec le verbe "déraciner" et le nom "arbre" dans la même fenêtre. Si on conserve les accords en nombre par exemple, le vocabulaire du modèle aura une entrée différente pour "vent" et pour "vents". Toutefois, la différence de sémantique entre ces entrées de lexiques est trop fines pour que nous nous y intéressions. Au contraire, différencier ces entrées pourrait diluer les associations entre le mot "vent" et d'autre éléments de ses contextes, en les partageant entre "vent" et "vents".

Nous utilisons la lemmatisation pour ramener les mots de nos corpus à des formes communes (voir 1.2.3), par exemple pour ramener "soufflé", "soufflés et "soufflées" à "souffler". Nous employons le LefffLemmatizer[2] pour le français et le lemmatiseur du modèle "*en_core_web_lg*"[3] pour l'anglais, tous deux à travers la bibliothèque Python spaCy. Le premier s'appuie sur une ressource lexicale du français publiée dans (Sagot, 2010), le second sur un modèle d'apprentissage entraîné sur un très grand volume de données tiré du web (articles de forum, post de blog, micro post de forum, etc). De plus, nous retirons le "s" final de mots qui ont déjà été rencontrés sans ce "s". Ce dernier

1. https://www.nltk.org/index.html
2. https://spacy.io/universe/project/spacy-lefff
3. https://github.com/explosion/spacy-models/releases/tag/en_core_web_lg-3.2.0

traitement est destiné à rattraper des erreurs de lemmatisation, ce processus n'étant pas précis à 100 %.

C'est dans le résultat de la lemmatisation que les mots vides sont recherchés pour être retirés. Une fois les mots ramenés à leur forme commune et les mots vides supprimés, nous ne conservons que les tweets contenant encore au moins trois mots. Finalement, nous remplaçons les nombres du corpus par le token `matchedNumber` et les heures par `matchedHour`. Le cas échéant, es caractères accentués sont remplacés par leur lettre sans accent, c'est-à-dire qu'un "à" est remplace par un "a" ou un "é" par un "e" dans le corpus.

4.2 Plongement des mots et documents

Pour construire des vecteurs représentant les tweets et leurs mots, nos utilisons le modèle Doc2Vec proposé dans (Le and Mikolov, 2014) et présenté dans la section 1.3.4.2. Nous choisissons ce modèle pour deux raisons principales. D'abord, c'est un modèle peu profond et il est donc facile et peu coûteux à entraîner de zéro. Son architecture légère lui permet de s'entraîner même sur une volumétrie de données restreinte. Dans le cadre de nos travaux, nous ne disposons pas de corpus faisant référence qui pourrait permettre d'entraîner un modèle pour détecter des évènements. Il nous faut donc un modèle qui soit facile à adapter d'un cas d'application à l'autre.

L'autre motivation derrière l'emploi de ce modèle est le fait que son architecture soit facile à appréhender. La finalité de nos travaux est de produire des descriptions compréhensibles pour un humain. Les représentations des documents et des mots sont à la base du clustering pour lequel nous allons construire les descriptions. Ainsi, si le modèle qui les construit est une boîte noire, il est difficile de pouvoir comprendre ce qui a amené des tweets à être groupés et ensuite décrits avec tel ou tel mot. Nous souhaitons donc éviter d'avoir recours à des modèles profonds dont il est difficile, ou pas encore possible, d'expliquer le fonctionnement.

Selon la sémantique distributionnelle présentée dans la section 1.3.1, la distance dans l'espace appris par le modèle correspond directement à une forme d'éloignement sémantique. Ainsi deux documents proches dans cette espace devraient avoir des sujets similaires.

Le modèle étant non supervisé, nous l'entraînons de zéro sur chacun de nos corpus pour obtenir les représentations. Nous paramétrons le modèle pour des vecteurs de taille 500, avec une fenêtre de taille 4, soit avec deux mots avants et deux après le mot cible d'une prédiction. Nous utilisons l'implantation proposée par Gensim 4.1.2 [4]. Nous avons fait différents essais de paramétrage, sans constater de variation notable au niveau du clustering. Le seul paramètre ayant eu un impact réellement significatif est la durée, mesurée en nombre d'époques, de l'entraînement. Une époque correspond au passage en revue de chaque élément du jeu d'entraînement. Nous explorons ce point en détail dans la section 6.1. Dans le cadre de ces expérimentations, l'entraînement dure 10 époques. Nous utilisons l'architecture PV-DM du modèle, montrée dans la figure 4.1, et dont nous détaillons le fonctionnement dans le chapitre 6.

4. `https://radimrehurek.com/gensim/`

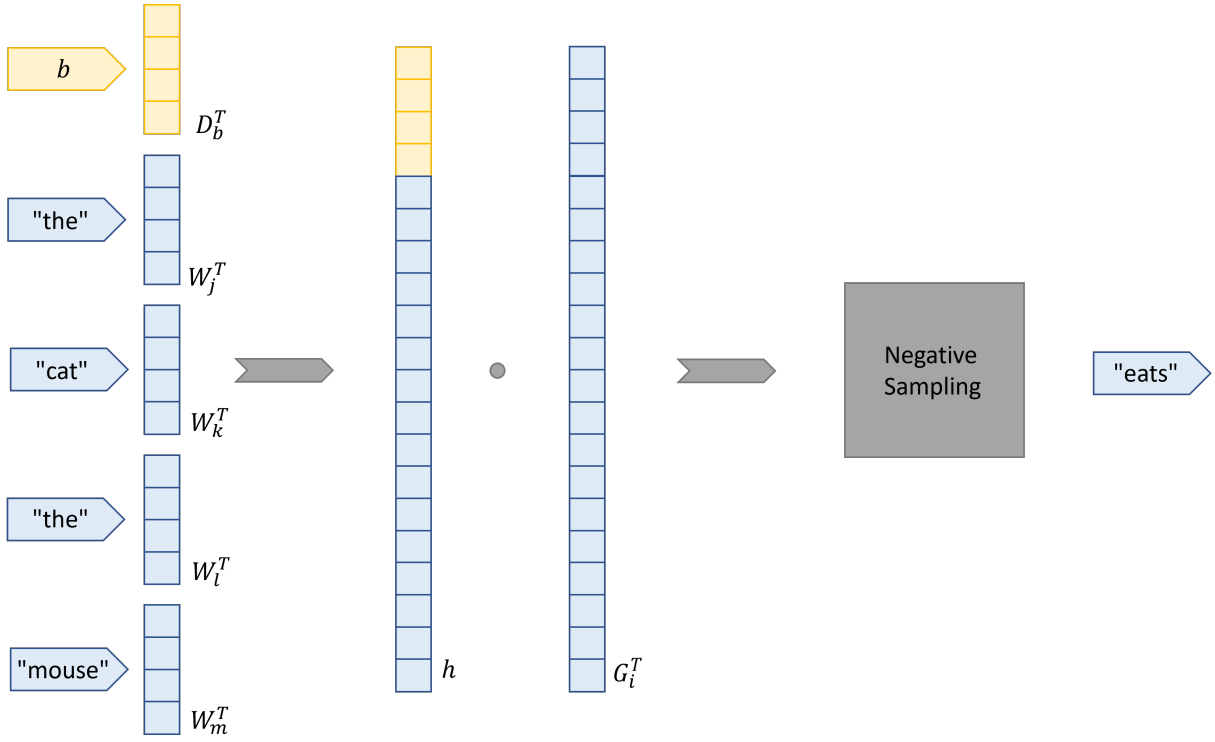


FIGURE 4.1 – Modèle Doc2Vec dans son architecture PV-DM

Les deux variantes, PV-DM et PV-DBoW, permettent selon les auteurs de l'article original d'obtenir des résultats similaires. Cela se vérifiant également dans la littérature traitant de l'évaluation ou de l'application de ces modèles, nous choisissons l'architecture la plus intuitive à comprendre pour un humain. En plus de cela, cette structure permet de garder distincte l'information issue de chaque entrée du modèle au cours de son raisonnement. Nous verrons en quoi ce point particulier revêt de l'intérêt dans le chapitre 6.

Nous avons également mené des expériences sur une représentation des documents avec des vecteurs de score TF-IDF. Les résultats obtenus avec cette méthodologie ont montré les limitations attendues. D'abord ils sont beaucoup plus long à clusteriser, les vecteurs de TF-IDF étant plus de dix plus fois plus longs que ceux obtenus avec Doc2Vec ($\approx$ 7500 contre 500). Par ailleurs, comme nous l'expliquons dans la prochaine section, le clustering que nous avons retenu est semi-aléatoire. Appliqués sur des vecteurs de score TF-IDF, son résultat est différent à chaque itération.

Chapitre 5

Des clusters de tweets à la description, évaluation des hypothèses de travail

Sommaire

Une fois les données préparées, nous pouvons leur appliquer les traitements qui vont permettre d'identifier les sous-évènements et de les décrire. Nos travaux abordent le problème de la description d'évènements à partir de tweets, captés dans un flux continu de données. Comme indiqué en introduction, nous faisons l'hypothèse que l'on peut décrire cet évènement en en décrivant les différentes manifestations visibles sur ce réseau social. Nous proposons de grouper les messages sur la base de leur sens pour en identifier des ensembles traitant du même sujet. Nous faisons l'hypothèse que ces groupements correspondent aux manifestations que nous recherchons. Ainsi, la mosaïque de leur description

constituera celle de l'évènement cible. Nous proposons de les décrire en extrayant les descripteurs directement depuis le contenu des groupements messages et d'employer un modèle déclaratif pour obtenir le meilleur ensemble de description possible.

Avant de traiter directement des données depuis un flux continu, nous avons mené des expérimentations pour tester la validité de nos hypothèses. Ce chapitre présente ces expérimentations, qui ont aboutit à une architecture de traitement de jeux de données statiques, et leurs résultats.

Étant donné que nous examinons des données Twitter, on peut naturellement penser aux hashtags pour décrire les données. Mais ils ne sont en réalité pas appropriés pour notre objectif. Ils sont principalement utilisés pour lier un tweet à une tendance en cours sur la plate-forme et améliorer sa visibilité. Il n'y a aucune garantie qu'un tweet contienne des hashtags ni qu'un hashtag soit représentatif du contenu des tweets. Il est donc plus approprié d'identifier les descripteurs à partir des tweets eux-mêmes.

La solution présentée dans ce chapitre et ses résultats ont fait l'objet d'une publication dans la conférence ICTAI de 2022 (Gracianne et al., 2022).

5.1 Architecture de l'approche proposée

L'objectif de ces travaux de thèse est d'obtenir une chaîne de traitement de bout-en-bout, ingérant des tweets et produisant des descriptions. Nous souhaitons donc éprouver nos hypothèses sur une chaîne qui soit également de bout-en-bout. Ainsi nous pourrons également identifier les éventuelles complications pouvant apparaître aux jonctions entres les composants de la chaîne.

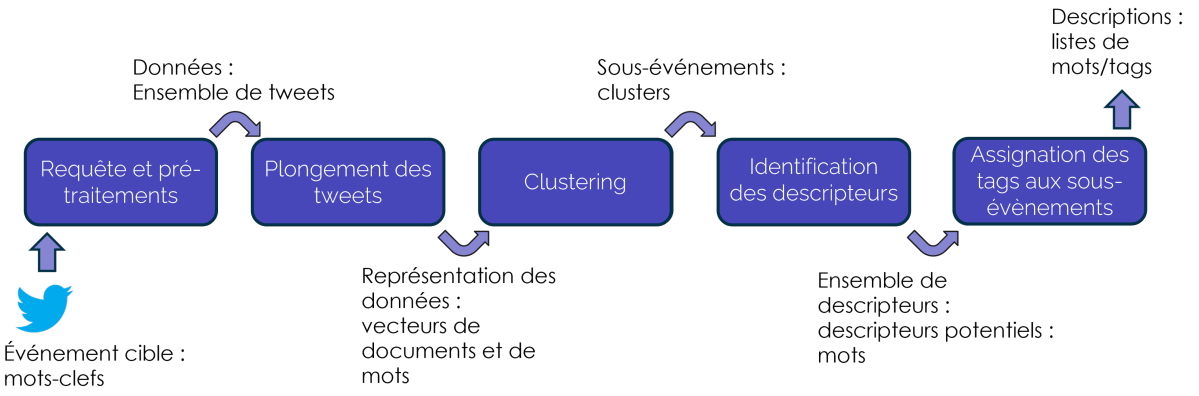


FIGURE 5.1 – Architecture bout-en-bout de description de clusters de tweets proposée pour éprouver les hypothèses des travaux de thèse (publiée dans (Gracianne et al., 2022))

L'architecture que nous proposons est celle représentée sur la figure 5.1. Les deux premières briques sont présentées en chapitre 4. Nous procédons à la capture des tweets via l'API de requétage de Twitter, pré-traitons les données afin d'obtenir des documents dépouillés de "stop words" et lemmatisés. Les nombres sont remplacés par le terme générique `matchedNumber`. Nous plongeons les documents ainsi que le vocabulaire de l'ensemble du corpus obtenu dans deux espaces de mêmes dimensions en utilisant Doc2Vec. Nous détaillons ici les composants suivants.

5.1.1 Clustering

L'algorithme de clustering utilisé est un composant central de notre approche, puisque c'est sur lui que repose l'identification des sous-évènements sous la forme d'ensembles de tweets.

Nous avons testé plusieurs stratégies de clustering pour évaluer ce qu'elles apportaient au clustering de documents vectorisés. Même sur des corpus relativement réduits comme ceux sur lesquels nous avons travaillé, il n'est pas possible de passer manuellement en revue les résultats de chaque exécution. Puisque K-Means construit des clusters sphériques, certaines de leurs caractéristiques conviennent pour les profiler et nous permettre de reconnaître ceux qui sont similaires d'une exécution à l'autre. Nous mesurons pour chaque cluster sa taille, la distance moyenne à son centroïde, ses mots les plus fréquents et son indice de silhouette (Rousseeuw, 1987). L'indice de silhouette d'un cluster est une métrique qui permet d'évaluer sa qualité selon deux critères : sa séparation avec les autres clusters et à quel point il est compact. Il est définit dans $[-1; 1]$ et se mesure à l'échelle d'une instance, pouvant donc être moyenné pour un cluster. Plus un cluster est compact et distinct des autres, plus son indice de silhouette tend vers 1. Pour un cluster $\mathcal{C}$ d'une partition $\mathcal{P}$, elle se calcule comme suit :

$$sil(\mathcal{C}) = \frac{1}{|\mathcal{C}|} \sum_{d \in \mathcal{C}} \frac{b(d) - a(d)}{max(a(d), b(d))} \tag{5.1}$$

où $a(d)$ est la distance moyenne entre le document d et les autres documents de son cluster soit :

$$a(d) = \frac{1}{|\mathcal{C}|} \sum_{d' \in \mathcal{C}, d' \neq d} dist(d, d')$$

et $b(d)$ est la distance moyenne entre d et les documents du cluster le plus proche de $\mathcal{C}$ soit :

$$b(d) = \min_{\mathcal{C}' \in \mathcal{P}, \mathcal{C}' \neq \mathcal{C}} \frac{1}{|\mathcal{C}'|} \sum_{d' \in \mathcal{C}'} dist(d, d')$$

Plusieurs possibilités sont envisageables pour clusteriser les documents. Comme nous l'avons exposé dans le chapitre 2, nous ne souhaitons pas employer de clustering conceptuel puisque ce type d'approche ne peut pas exploiter des représentations neuronales. Il nous est donc possible d'employer des méthodologies de clustering plat ou hiérarchique, que nous avons testées pour identifier la meilleure. Notons qu'avant même ces tests, les approches hiérarchiques retiennent moins notre intérêt. Cela est du au fait que la structure arborescente qu'elles identifient ne revêt pas un intérêt particulier dans le cadre de nos travaux. Nous savons que les différentes manifestations de l'évènement cible visibles dans les données sont reliées entre elles du fait de la requête qui a permis d'obtenir les données. Par construction, ces tweets sont liés ontologiquement. Mais il n'y a aucune garantie que tout le contenu récupéré par cette requête soit lié, que ce soit d'une manière hiérarchique ou d'une autre, à la cible du fait de la nature de Twitter. De fait, nous ne cherchons pas de hiérarchie entre ces manifestations. La structure identifiée par un clustering hiérarchique ne revêt donc d'intérêt particulier pour nos travaux. Ainsi seuls les clusters identifiés nous intéressent, ce qui nous amène à considérer le résultat d'un clustering hiérarchique de la

même manière qu'un clustering plat. Les tableaux suivants présentent les profils types des clusters obtenus.

critère / cluster	taille	distance moyenne au centroïde	silhouette moyenne	mots les plus fréquents
0	1476	0.1654	0.1502	alpes-maritime, apre, #alpesmaritime, personne, disparu, tout, passage, pompier, village, sinistre
1	1683	0.13	0.1489	c'est, tout, aller, plus, faire, cest, bien, quand, vent, meme
2	631	0.1795	0.3389	tout, matchedNumber, plus, c'est, vent, faire, apre, comme, bien, temp
3	2743	0.1189	0.2214	tout, apre, c'est, plus, dune, faire, aller, ainsi, bout, trois
4	1705	0.1537	0.2496	tout, plus, c'est, faire, vent, matchedNumber, aller, apre, comme, jour
5	3538	0.1114	0.2297	aller, c'est, tout, cest, faire, plus, vent, apre, bretagne, bien
6	1776	0.1554	0.1692	alpes-maritime, apre, personne, disparu, passage, france, corps, mort, franceinfo, #alpesmaritime
7	771	0.1845	0.1929	vent, matchedNumber, kilometre, rafale, nuit, meteo, #bretagne, fort, bretagne, vendredi
8	4545	0.1007	0.2268	alpes-maritime, tout, apre, aller, vent, france, bretagne, degat, faire, vigilance
9	1148	0.1625	0.1362	vent, vigilance, meteo, bretagne, france, rouge, morbihan, orange, alerte, alpes-maritime

TABLEAU 5.1 – Exemple de profil de clustering obtenu sur le jeu de données sur la tempête Alex

critère / cluster	taille	distance moyenne au centroïde	silhouette moyenne	mots les plus fréquents
0	4137	0.086	0.28	russia, russian, putin, stop, people, need, nato, invasion, #stopputin, support
1	1726	0.112	0.206	putin, russia, would, like, people, think, world, trump, know, make
2	2994	0.102	0.25	russia, putin, people, would, matchedNumber, world, like, country, make, russian
3	528	0.157	0.381	russia, russian, matchedNumber, people, putin, make, like, would, military, support
4	4096	0.088	0.322	russia, putin, people, nato, russian, world, would, support, like, invasion
5	994	0.162	0.163	people, need, stop, russian, defend, russia, putin, help, weapon, #putin
6	632	0.167	0.206	nato, russia, russian, potus, close, security, #stopputin, #stoprussia, council, stop
7	1846	0.117	0.32	russia, putin, matchedNumber, people, russian, would, like, world, country, think
8	17	0.341	0.619	mikenieve, rahmmagick, queens_dude, 0oty_mac, woodshed_1914, eepdllc, j1gg__, jedigollum01, trace_avp, jaelin_taylor
9	1257	0.142	0.157	russian, russia, nato, stop, putin, people, #russia, need, invasion, help

TABLEAU 5.2 – Exemple de profil de clustering obtenu sur le jeu de données sur la guerre Russio-Ukrainienne

Les profils présentés dans les tableaux 5.1 et 5.2 nous permettent d'évaluer la stabilité des résultats. Ces deux tableaux montrent les profils des clusters obtenus avec K-Means. Parmi les algorithmes de clustering plat existants, nous l'avons choisi puisqu'il est le plus simple à mettre en application et que son fonctionnement est très robuste. Pour les approches hiérarchiques, nous avons testé des stratégies de maximum, average et Ward linkage (voir section 2.1.4). Les trois stratégies offrent des résultats très similaires sur nos données et aboutissent à des profils de clusters qui sont très proches de ceux obtenus avec K-Means. Ces premiers résultats ont montré que l'approche hiérarchique ne permet pas d'obtenir de meilleurs résultats qu'un K-Means, et nous l'avons donc écartée.

K-Means est un bon candidat d'algorithme de clustering. Sa stratégie de minimisation des distances au centroïde correspond parfaitement avec notre objectif d'identifier des clusters denses. Plus un cluster est compact dans un espace de représentation comme celui appris par Doc2Vec, plus les messages qu'il contient devraient avoir un sujet similaire. Un point important qui reste à prendre en compte est l'initialisation de K-Means. Cette dernière est semi-aléatoire, car nous utilisons l'initialisation K-Means++ (Arthur and Vassilvitskii, 2007). Plutôt que de choisir les premiers centroïdes complètement aléatoirement, on le fait de sorte à maximiser la distance qui les sépare. Il reste néanmoins une part d'aléatoire dans cette sélection, et il faut vérifier son impact sur les clusters obtenus. Pour cette validation, nous avons répété le clustering sur les mêmes données sur 20 séries de 200 exécutions. Même avec l'initialisation K-Means ++, les clusters obtenus avec les vecteurs de score TF-IDF sont très différents à chaque exécution. Avec les plongements neuronaux, on obtient des clusters et des profils presque identiques à chaque itération. Cette stabilité nous encourage à retenir Doc2Vec et K-Means pour représenter et cluster-iser nos documents. Ce constat est appuyé par les conclusions de l'étude (Curiskis et al., 2020). Ses auteurs montrent que cette combinaison (Doc2Vec avec K-Means) est utile pour la tâche d'extraction d'information.

Dans le tableau 5.1, on peut constater que les clusters obtenus ont tous des profils très similaires. En explorant les données, il est apparu que cela est du au fait que les mots les plus fréquents présentés ici sont présents dans presque toutes les conversations au sujet de la tempête dans notre corpus. La fréquence seule ne suffit donc pas à identifier des mots différents du profil d'un cluster à celui d'un autre. Nous pouvons aussi constater que les indices de silhouettes moyens des clusters ne sont pas notablement élevés. Toutefois, il est connu que ce type de métrique ne permet pas d'affirmer ou d'infirmer que le clustering a bien permis de capturer de l'information. Il permet juste d'estimer si les clusters sont clairement distingués les uns des autres tout en étant compacts. Ce n'est donc pas surprenant de voir des valeurs comme celles-ci avec des données venant de Twitter. Les données venant de cette plate-forme présentent souvent un contenu diffu ou difficile à isoler du reste, ce dernier point étant accentué par la petite taille des messages.

Paramétrage de K Identifier le nombre de clusters, soit de sous-évènements, est une difficulté pour ces expérimentations. Nous n'utilisons aucune connaissance a priori de la cible pour simuler la découverte de l'évènement dans un jeu de données statique.

En règle générale, il est possible d'identifier un nombre optimal de clusters à chercher en employant la méthode du coude par rapport à un critère de qualité de la partition dans son ensemble. Nous avons donc essayé de mettre en place cette méthode. Nous avons d'abord essayé d'utiliser l'indice de Davies-Bouldin (Davies and Bouldin, 1979) comme

critère à optimiser. C'est un indice similaire à la silhouette, mesurant la qualité d'une partition à la séparation et la compacité de ses clusters. Il se calcule comme suit :

$$DB(\mathcal{P}) = \frac{1}{K} \sum_{\mathcal{C} \in \mathcal{P}} \max_{\mathcal{C}' \neq \mathcal{C}} \frac{\delta_{\mathcal{C}} + \delta_{\mathcal{C}'}}{dist(\mu_{\mathcal{C}}, \mu_{\mathcal{C}'})} \tag{5.2}$$

où $\delta_{\mathcal{C}}$ est la distance moyenne entre les documents de $\mathcal{C}$ et son centroïde $\mu_{\mathcal{C}}$. L'indice de Davies-Bouldin pour une partition est défini sur $[1, +\infty]$, avec 1 pour la meilleure classification. Cela nous a permis d'obtenir le résultat suivant pour le jeu de données sur la guerre Russo-Ukrainienne, très similaire à celui obtenu pour la tempête Alex.

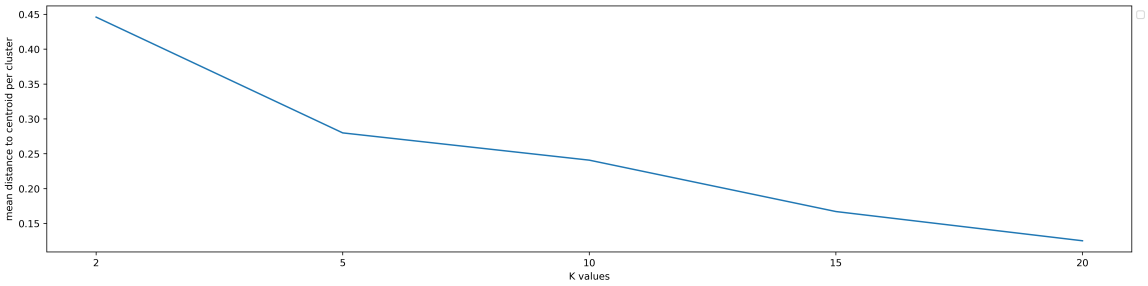


FIGURE 5.2 – Indice de Davies-Bouldin obtenu avec différentes valeurs de K, avec K-Means exécuté sur le corpus sur la guerre Russo-Ukrainienne

La méthode du Coude permet d'identifier visuellement un paramètre optimal par rapport à un critère. Le "coude" est la forme adoptée par la courbe de la valeur de ce critère en fonction du paramètre. Elle indique la valeur pour laquelle l'augmentation ou la diminution de la valeur du paramètre ne permet plus d'améliorer significativement le critère. Dans la figure 5.2, où le meilleur indice de Davies-Bouldin est atteint pour $K = 2$, il est difficile d'identifier un coude clair. On peut voir un coude pour $K = 5$, mais l'indice a une valeur similaire pour $K = 10$. Un indice comme celui de Davies-Bouldin s'appuie sur la mesure de la dispersion des clusters dans la partition et de leur compacité. Comme la combinaison de ces deux mesures ne permet pas d'identifier un coude clair, nous les mesurons séparément.

Nous mesurons la compacité d'une partition avec la somme de ses inerties intra-cluster, calculées avec la distance des documents à leur centroïde soit :

$$intra(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} \sum_{d \in \mathcal{C}} dist(\mu_{\mathcal{C}}, d)$$

Nous mesurons sa dispersion comme la somme des distances entre les centroïdes des clusters et le centroïde de la partition soit :

$$inter(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} dist(\mu_{\mathcal{P}}, \mu_{\mathcal{C}})$$

où $\mu_{\mathcal{P}}$ est le point obtenu en faisant la moyenne de tous les points du corpus.

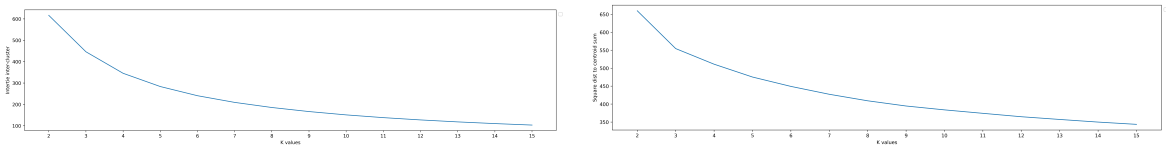


FIGURE 5.3 – Inerties inter et intra-cluster en fonction de K pour une partition obtenue avec K-Means sur les données de la tempête Alex

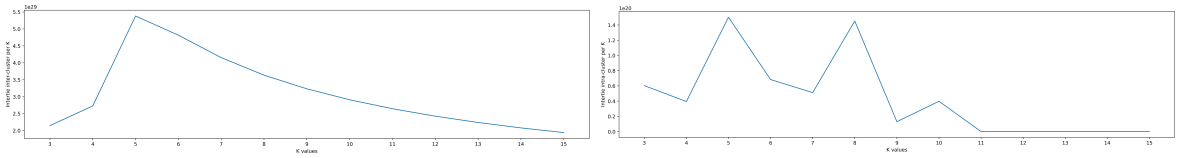


FIGURE 5.4 – Inerties inter et intra-cluster en fonction de K pour une partition obtenue avec K-Means sur les données de la guerre Russo-Ukrainienne

Ces mesures ne semblent toutefois pas convenir pour nos données puisque le comportement de leur valeur n'est pas cohérent avec ce qu'elles décrivent. L'inertie inter-cluster, présentée sur le graphique de gauche dans les figures 5.3 et 5.4, est censée augmenter quand K augmente. Pour les deux jeux de données, elle fait l'inverse. L'inertie intra-cluster, présentée sur le graphique de droite dans les figures 5.3 et 5.4, est censée diminuer quand K augmente. Si elle suit ce comportement pour les données sur la tempête Alex, elle est erratique pour les données sur la guerre en Ukraine.

À défaut de trouver un critère à optimiser qui convienne pour nos données, nous identifions la valeur de K empiriquement. Toutefois, cette lacune est appelée à être adressée naturellement avec une approche incrémentale qui détecte le nombre de clusters dans les données. Au fil des tests et d'estimation manuelle du contenu des clusters, 10 est apparu comme un juste milieu entre de l'information trop dispersée et des informations trop compactées.

5.1.2 Sélection des descripteurs

Une fois les clusters de tweets construits, il faut identifier les mots qui serviront à les décrire et nous avons testé plusieurs critères pour les trouver.

Les tags doivent permettre de décrire le contenu de leur cluster et d'en identifier la spécificité par rapport aux autres. Comme nous l'avons vu dans les profils de clusters, la fréquence des mots seule semble ne pas suffire pour cet objectif, et il nous faut vérifier cette hypothèse. Pour faire mieux, nous proposons d'adapter le score TF-IDF aux données issues de Twitter. Nous testons également si l'intégration d'un critère de couverture de l'espace de représentation des mots permet d'identifier de meilleurs descripteurs.

L'objectif de cette étape est d'identifier un ensemble de descripteurs candidats qui est commun à tous les clusters. Ainsi un modèle d'assignation pourra l'utiliser pour construire les meilleures descriptions à l'échelle du corpus. Pour ce faire, nous identifions l'ensemble des meilleurs candidats pour chaque cluster et fusionnons ensuite ces ensembles. Nous présentons dans cette section les différents types de sélections que nous proposons. Il

s'agit des sélections par fréquence, par score DF-IDF et une sélection hybridant le score DF-IDF avec une exploration de l'espace de représentation des mots.

5.1.2.1 Fréquence

Cette sélection a pour but de vérifier expérimentalement que la fréquence seule ne permet pas d'identifier un bon ensemble de descripteurs candidats pour la partition. Dans chaque cluster, les mots sont classés selon leur fréquence brute, soit selon la somme du nombre de leurs occurrences dans les tweets de ce cluster.

Nous retenons la liste des 20 mots les plus fréquents par cluster. Ce nombre, 20, est utilisé car la taille d'un tweet est limitée à 280 caractères. Au maximum, un tweet peut donc compter entre 40 et 50 mots. Une fois les pré-traitements effectués, un tweet comptera au maximum une vingtaine de mots. Retenir 20 mots par cluster revient donc à isoler, comme tags candidats qui en sont issus, l'équivalent en nombre de mots de la longueur d'un tweet quand il est traité par notre approche.

Ces listes sont agrégées et les duplications sont retirées du résultat de cette fusion pour obtenir l'ensemble des tags candidats de la partition. Si un mot apparaît plusieurs fois dans la liste de cette fusion, on en laisse une occurrence unique.

La redondance parmi les candidats n'a en effet pas d'intérêt dans le cadre de nos travaux. Nous choisissons de contrôler la redondance au sein des descriptions via le modèle d'assignation. Ainsi, la sélection de candidat doit seulement permettre d'obtenir les matériaux nécessaires pour décrire les différentes manifestations de l'évènement cible. Faire le choix d'utiliser un mot plusieurs fois ne doit pas être fait à ce stade.

5.1.2.2 Score DF-IDF

Le score TF-IDF est un outil très courant en extraction d'information depuis des données textuelles. Nous l'avons présenté en section 1.2.3. Toutefois, il n'a pas été pensé pour être appliqué sur des documents courts. Nous proposons d'adapter ce score à ce cadre, d'une manière similaire à ce qui a été fait ailleurs dans la littérature. Dans (Weng et al., 2011) par exemple, le score TF-IDF est adapté pour pouvoir être appliqué à des tweets. Pour cela, l'opérande *TF* pour "*Term Frequency*" est modifiée pour devenir *DF* pour "*Document Frequency*". Dans ces travaux, les auteurs travaillent sur des fenêtres temporelles glissantes de collecte de tweets. Le *DF* mesure ainsi le nombre de tweet contenant le mot dans la fenêtre étudiée. Les auteurs utilisent ensuite ce score pour traduire chaque mot en composante d'un signal représentant le tweet, et traitent ces derniers via des outils de traitement de signal.

Nous nous inspirons de cette proposition et l'adaptons pour travailler sur des partitions de tweets plutôt que sur des fenêtres temporelles. Le score DF-IDF que nous proposons se calcule pour un mot w par rapport à un cluster $\mathcal{C}$ d'une partition $\mathcal{P}$ comme suit :

$$\text{DF-IDF}(w, \mathcal{C}, \mathcal{P}) = \frac{N_w(\mathcal{C})}{|\mathcal{C}|} \log(\frac{N(\mathcal{P})}{N_w(\mathcal{P})}) \tag{5.3}$$

où $N_w(\mathcal{C})$ est le nombre de tweet dans $\mathcal{C}$ contenant w, $|\mathcal{C}|$ est le nombre de tweet dans $\mathcal{C}$, $N(\mathcal{P})$ est le nombre de tweet dans toute la partition $\mathcal{P}$ et $N_w(\mathcal{P})$ est le nombre de tweet dans $\mathcal{P}$ contenant w.

L'adaptation du TF en DF, l'opérande de gauche du score, permet de pallier la limitation du TF-IDF induite par la mesure de la répétition d'un terme au sein d'un document. Dans un message court, le TF d'un terme est presque toujours égal à 1. Il faut donc considérer le nombre de fois où un mot apparaît dans un ensemble donné, ici le cluster, pour que le décompte de cette répétition ait un sens. Ainsi, les mots apparaissant dans une plus grande proportion des messages de leur cluster auront une plus grande valeur de DF. Nous pondérons le DF par la taille du cluster pour lequel le score d'un mot est calculé pour éviter que les mots aient, mécaniquement, de meilleurs scores dans les clusters plus grands.

La logique de l'IDF, l'opérande de droite du score, reste la même. Son objectif est de valoriser les mots rares à l'échelle du corpus. Ainsi, les mots apparaissant dans tous les tweets de la partition auront une moins grande valeur d'IDF.

La formulation du DF-IDF que nous proposons permet ainsi d'identifier au sein d'un cluster les mots qui en sont les plus discriminants. Elle permet en outre de capturer numériquement de l'information liée à la topographie de la partition.

Nous nous servons de ce score pour classer les mots dans chaque cluster. Comme pour la fréquence brute, nous retenons la liste des 20 mots de plus haut DF-IDF par cluster. L'ensemble des tags candidats est obtenu par la fusion de ces listes, de laquelle on a éliminé les répétitions.

5.1.2.3 Sélection hybride DF-IDF - FPF

Dans l'idéal, notre approche décrirait toutes les manifestations de l'évènement cible présentes dans les données. Un axe possible pour poursuivre cet objectif est d'essayer que les tags candidats couvrent le plus possible l'espace de représentation des mots du corpus. L'ensemble de tags ainsi obtenu permettrait de couvrir une plus grande partie des objets dont il est possible de parler avec les mots du corpus. Il faut toutefois nuancer cet objectif, en gardant à l'esprit que toutes les données du corpus ne sont pas pertinentes au regard de notre objectif. D'une part, le corpus peut contenir des messages qui ne sont pas liés à la cible. D'une autre, même le contenu lié à l'évènement peut être trop bruité pour être utile à la description d'une de ses manifestations.

Nous expérimentons donc une sélection de tags candidats qui identifie les mots les plus couvrants de leur espace de représentation. Ceux-ci ne sont retenus que s'ils ont un DF-IDF dépassant un seuil, déterminé par rapport aux autres scores DF-IDF du cluster.

Cette sélection est basée sur l'algorithme *FPF* pour *Farthest Point First*, proposé dans (Gonzalez, 1985). Celui-ci cherche à identifier des représentants, ou têtes, parmi un ensemble de point. La première tête est le point qui maximise la distance à tous les autres points. Ensuite, l'algorithme cherche à chaque étape le point le plus éloigné de sa tête courante. Il devient à son tour une tête, et chaque autre point plus proche de lui que de sa tête courante lui est raccroché. Nous établissons un seuil de DF-IDF par cluster pour qu'un mot fasse partie de ses représentants possibles. La procédure hybride que nous proposons d'utiliser est présentée dans l'algorithme 12.

Nous recherchons avec cette sélection hybride 20 représentants par cluster. Comme pour les autres, nous établissons l'ensemble des tags candidats avec la fusion des listes de candidats par cluster dont nous supprimons les duplications. Le seuil de DF-IDF à surpasser pour pouvoir faire partie des tags candidats est le score DF-IDF médian du cluster. On s'assure ainsi que les mots issus de cette sélection font partie des plus discriminants

Algorithme 12: Sélection de tags candidat hybride DF-IDF - FPF

input: la partition $\mathcal{P}$, le nombre k de représentant recherché par cluster

$tags \leftarrow [\,]$
Pour chaque *cluster* $\mathcal{C} \in \mathcal{P}$ **faire**
  $n \leftarrow [\,]$ `/* n va contenir les mots dépassant le seuil de DF-IDF nécessaire pour qu'un mot puisse être sélectionné comme tag candidat dans` $\mathcal{C}$ `*/`
  $tags_{\mathcal{C}} \leftarrow [\,]$
  Pour chaque *mot* $w \in \mathcal{C}$ **faire**
    Si *DF-IDF*$(w, \mathcal{C}, \mathcal{P}) >$ *DF-IDF médian de* $\mathcal{C}$ **alors** ajouter w dans n;
    `/* ici, on compare le score DF-IDF médian de` w `dans` $\mathcal{C}$ `au DF-IDF médian de` $\mathcal{C}$ `*/`
  $t \leftarrow argmax_{w \in \mathcal{C}} \sum_{w' \in \mathcal{C}, w' \neq w} dist(w, w')$ `/* la première tête dans le cluster est le mot qui maximise la distance aux autres mots du cluster */`
  ajouter t à $tags_{\mathcal{C}}$
  affecter tous les points de $\mathcal{C}$ à la tête t
  Pour i *de 1 à* $k-1$ **faire**
    $nt \leftarrow argmax_{w \in n}\ dist(w, tete(w))$ `/* la nouvelle tête est le mot le plus éloigné de sa tête actuelle */`
    affecter tous les points plus proche de nt que de leur tête actuelle à nt
    ajouter nt à $tags_{\mathcal{C}}$
  concaténer $tags_{\mathcal{C}}$ à $tags$
retourner $tags$

du cluster où ils ont été identifiés. L'algorithme *FPF* permet quant à lui de garantir que les tags candidats couvrent une plus grande partie de l'espace sémantique du cluster.

5.1.3 Assignation des descripteurs aux clusters

Nous proposons de formuler le modèle d'assignation des tags aux descriptions des K clusters d'une partition $\mathcal{P}$ dans un formalisme déclaratif. Pour cela, nous utilisons le modèle de PLNE suivant. Il prend en entrée :

— les m candidats descripteurs issus de la sélection de tags employée ;
— une matrice P de taille $K \times m$ où P_{ij} est le score DF-IDF du candidat j dans le cluster i ;
— pour chaque cluster $\mathcal{C} \in \mathcal{P}$, une matrice $T^{\mathcal{C}}$ de taille $|\mathcal{C}| \times m$ où $T^{\mathcal{C}}_{ij}$ vaut 1 si le candidat j apparaît dans le tweet i du cluster $\mathcal{C}$ et 0 sinon.

Pour représenter l'assignation d'un tag à une description, nous définissons une matrice de variables binaires X de taille $K \times m$ où X_{ij} vaut 1 si le tag j est assigné à la description du cluster i et 0 sinon.

Nous souhaitons obtenir les descriptions les plus représentatives des clusters que l'on peut obtenir à partir des tags candidats identifiés par la sélection utilisée. L'objectif du modèle sera donc la maximisation de la discriminance des descriptions obtenues, soit la maximisation de leur score DF-IDF :

$$\texttt{maximiser} \sum_{i=1}^{K} \sum_{j=1}^{m} X_{ij} P_{ij}$$

Pour guider la construction des descriptions et garantir leur qualité, nous définissons les contraintes suivantes :

— **Non vide.** Une description doit contenir au moins un mot :

$$\forall i \in [1, K], \sum_{j=1}^{m} X_{ij} \geq 1 \tag{5.4}$$

Une description vide n'apporte aucune information sur son cluster et donc aucune information sur l'évènement cible. Elle n'a donc pas d'intérêt pour nos travaux et nous interdisons donc cette éventualité.

— **Diversité.** Un même tag peut être utilisé dans au plus β descriptions :

$$\forall j \in [1, m], \sum_{i=1}^{K} X_{ij} \leq \beta \tag{5.5}$$

La diversité au sein des descripteurs est un critère de qualité très répandu dans le domaine du résumé automatique (voir section 3.4). Cette contrainte permet de garantir et de contrôler cette diversité. La diversité est maximale si $\beta = 1$, soit si chaque descripteur ne peut apparaître que dans une description. Augmenter la valeur de β permet d'autoriser un certain chevauchement des descriptions.

— **Taille maximum.** Une description doit contenir au plus t_{max} mots :

$$\forall i \in [1, K], \sum_{j=1}^{m} X_{ij} \leq t_{max} \tag{5.6}$$

Notre objectif est de produire des descriptions faciles à comprendre pour un humain. Plus une description est longue, plus elle difficile à comprendre. En effet, plus de tags peuvent diluer le sens d'une description ou lui faire couvrir plusieurs aspects de son cluster. Cela fera en outre chercher à l'utilisateur plus de combinaison de mots pour comprendre ce dont traite le cluster. Nous limitons donc le nombre maximum de tag que peut compter une description.

— **Score non nul.** Un tag doit avoir un score DF-IDF supérieur à 0 dans le cluster de la description à laquelle il est assigné :

$$\forall i \in [1, K], \forall j \in [1, m], P_{ij} = 0 \implies X_{ij} = 0 \tag{5.7}$$

Dans un cas non favorable où le problème d'assignation est particulièrement difficile à résoudre, par exemple si la sélection de tags a identifié des candidats descripteurs peu diversifiés, le problème suivant peut se poser. Les interactions de la contrainte de non vacuité et de celle de diversité peuvent amener le modèle à assigner un tag à une description alors que celui-ci a un score DF-IDF nul das le cluster concerné. Comme cela aboutirait à une description hors-sujet, nous interdisons cette éventualité.

— **Couverture d'occurrences.** La somme des occurrences d'un tag d'une description doit être supérieure à une proportion α de la taille du cluster concerné :

$$\forall \mathcal{C} \in \mathcal{P}, \sum_{j=1}^{m} \sum_{i=1}^{|\mathcal{C}|} T_{ij}^{\mathcal{C}} X_{\mathcal{C}j} \geq \alpha |\mathcal{C}| \tag{5.8}$$

Pour être de bonne qualité, nous estimons qu'une description doit couvrir une proportion significative des tweets du cluster qu'elle décrit. Nous formulons cette contrainte avec des nombres entiers dans l'équation 5.8. Notons qu'écrite de la sorte, cette contrainte compte le nombre de fois où un tweet contient un tag. Il ne s'agit pas de "couverture" des documents au sens habituel du mot, mais cette formulation l'approxime efficacement. Notons également que le paramètre α peut avoir une valeur différente pour chaque cluster.

Nous avons utilisé l'interface Python du solutionneur Gurobi 9.1.12 [1] pour implanter le modèle. Un solutionneur est un logiciel qui explore les différentes solutions atteignables par le modèle et identifie la meilleure par rapport à l'objectif et aux contraintes du modèle.

5.2 Évaluation des descriptions

Évaluer de manière automatique les descriptions que nous obtenons est une tâche difficile. À notre connaissance, il n'est pas possible sans référence construite à la main d'estimer automatiquement si une description reflète la sémantique de son cluster. De même il n'est pas possible d'évaluer entièrement automatiquement la qualité linguistique d'une description.

En modélisation de sujet, deux métriques très utilisées pour évaluer la qualité des mots des sujets sont la diversité et la cohérence de sujet. Avec notre modèle d'assignation de tag, la diversité est garantie par la contrainte 5.5. La cohérence n'est pas une mesure

1. https ://www.gurobi.com/documentation/9.1

appropriée dans notre cas puisqu'elle se base sur les co-occurrences des mots des sujets dans de mêmes documents. Comme nous travaillons sur des documents courts, sélectionner des tags selon ces co-occurrences limiterait leur diversité.

Nous proposons donc deux métriques pour mesurer la qualité de nos descriptions. Comme il n'est pas possible d'évaluer la qualité absolue de ces dernières, nos métriques ont pour objectif de les comparer entre elles. L'objectif est de mesurer si les descriptions obtenues pourraient être meilleures.

Ces critères d'évaluation sont l'**IR** pour l'Importance Relative d'une description et la **SPD** pour la Somme Pondérée des Distances.

5.2.1 IR

Notre objectif est de décrire les manifestations de l'évènement présentes dans les données de sorte à pouvoir les distinguer et en comprendre la teneur. Nous établissons donc comme mesure de pertinence de l'assignation d'un tag à un cluster sa propension à le distinguer des autres d'une part, et d'être d'autre part représentatif de son contenu. C'est ce que mesure le score DF-IDF et nous proposons une métrique qui se base sur lui.

L'IR d'une description se calcule pour une description $e_{\mathcal{C}}$ par rapport à un cluster $\mathcal{C}$ appartenant à une partition $\mathcal{P}$ comme suit :

$$IR(e_{\mathcal{C}}, \mathcal{C}, \mathcal{P}) = \sum_{w \in e_{\mathcal{C}}} \text{DF-IDF}(w, \mathcal{C}, \mathcal{P}) \tag{5.9}$$

L'objectif de cette métrique est de mesurer à quel point le modèle d'assignation de tag a attribué les tags aux clusters pour lesquels les tags étaient les plus discriminants. Ainsi, une haute valeur d'IR n'indique pas qu'une description soit bonne dans l'absolu. Il faut comparer les valeurs d'IR d'un ensemble de descriptions pour évaluer si le modèle a construit les meilleures qu'il pouvait.

Une valeur d'IR pour une description $e_{\mathcal{C}}$ sera "bonne" si elle est significativement plus élevée pour le cluster $\mathcal{C}$ que pour les autres clusters de $\mathcal{P}$. En effet, cela indique que la description dans son ensemble est significativement plus discriminante pour son cluster que pour les autres.

À l'opposé, une valeur d'IR d'une description qui serait plus élevée pour un autre cluster que pour celui qu'elle décrit est une "mauvaise" valeur. Elle indiquerait une description dont les mots seraient plus appropriés pour identifier cet autre cluster que celui auquel elle attribuée. Ce cas de figure peut s'expliquer de plusieurs façons.

D'abord, la description peut simplement avoir mal été construite. Comme le modèle d'assignation de tags optimise un score de pertinence des descriptions, cela limite la variété des causes d'un tel cas de figure. Une description "mal construite" pourrait indiquer qu'une ou plusieurs des contraintes du modèle ne sont pas appropriées pour les données traitées. Il pourrait par exemple choisir un mot couvrant efficacement un cluster alors qu'il a un score plus élevé dans un autre cluster, où il est moins répandu.

Le modèle peut également être forcé d'attribuer des mots inadéquats à des descriptions s'il ne dispose pas d'un ensemble de candidats assez varié. Il pourrait alors être forcé d'attribuer un tag à la description d'un cluster pour qu'elle ne soit pas vide alors, ce tag ayant pourtant un meilleur score pour un autre cluster. Cette éventualité indiquerait que la sélection de candidats employée ne conviendrait pas à son cas d'application.

Enfin, une valeur d'IR plus élevée pour un autre cluster que celui auquel est attribué la description peut indiquer une difficulté de partitionnement. En effet, l'objectif du modèle est de maximiser la somme du score de pertinence des descriptions, pas de maximiser le score de chaque description. Optimiser cette somme ne garantit ainsi pas d'obtenir le meilleur score pour chaque description. Les tags peuvent avoir des scores similaires d'un cluster à l'autre s'il y a au sein de la partition des clusters difficiles à distinguer. Ainsi une description peut obtenir un meilleur score de pertinence pour un autre cluster que pour le sien.

L'importance relative donne de ce fait un aperçu de la séparation de l'information contenue dans les clusters. En effet, les descriptions obtenues peuvent avoir des valeurs d'IR similaires pour tous les clusters. Cela signifie que le modèle d'assignation n'a pas pu identifier de tags significativement plus discriminants pour un cluster que pour les autres. Par construction du score DF-IDF, cette situation indiquerait que les clusters obtenus ne seraient pas clairement distincts.

5.2.2 SPD

Pour qu'une description soit compréhensible, il faut que les éléments qui la composent soient cohérents. Toutefois, la cohérence sémantique que nous pouvons mesurer à travers la distance dans l'espace de représentation des mots n'a pas de valeur que l'on puisse interpréter dans l'absolu. De fait, nous proposons de vérifier que les mots d'une description soient plus proches entre eux que des mots d'une autre description. C'est ce que mesure la métrique SPD que nous proposons.

Elle se calcule pour une description $e_{\mathcal{C}}$ par rapport à une autre notée $e_{\mathcal{C}'}$. Pour cela, nous définissons d'abord une mesure S entre les deux descriptions telle que :

$$S(e_{\mathcal{C}}, e_{\mathcal{C}'}) = \frac{\sum_{w \in e_{\mathcal{C}}} \sum_{w' \in e_{\mathcal{C}'}} dist(w, w')}{|e_{\mathcal{C}}||e_{\mathcal{C}'}|} \tag{5.10}$$

S varie entre 0 et $+\infty$. Plus les distances entre les mots de $e_{\mathcal{C}}$ et $e_{\mathcal{C}'}$ sont petites, plus la valeur de $S(e_{\mathcal{C}}, e_{\mathcal{C}'})$ est basse. On peut donc se servir de la mesure S pour évaluer l'ensemble des descriptions obtenu. En effet, quel que soit le cas de figure, on attend $S(e_{\mathcal{C}}, e_{\mathcal{C}}) < S(e_{\mathcal{C}}, e_{\mathcal{C}'})\ \forall\ \mathcal{C}, \mathcal{C}' \in \mathcal{P}$. C'est-à-dire que l'on attend que les mots d'une description soient plus poches entre eux que proches des mots d'une autre description. Formulé différemment, qu'une description soit plus cohérente avec elle-même qu'avec une autre description.

Intuitivement, on pourrait s'attendre à ce que cela soit toujours le cas. Notons toutefois que la distance utilisée pour calculer la mesure S se mesure entre les vecteurs de mots de l'espace de représentation. Ces derniers sont les mêmes pour tous les documents. La représentation $\vec{w}$ d'une occurrence du mot w dans le document d est aussi $\vec{w}$ pour une occurrence dans un autre document. Ainsi, un mot peut être choisi pour décrire les documents d'un cluster alors qu'il apparaît dans ceux d'un autre cluster. Doc2Vec déduit les représentations des mots des contextes dans lesquels ils apparaissent. Alors un mot w peut tout à fait être plus proche des mots des documents d'un autre cluster que de ceux du cluster qu'il décrit. C'est notamment le cas des mots qui apparaissent régulièrement dans certains contextes, et ponctuellement dans d'autres.

Pour faciliter la lecture des scores de SPD pour un ensemble de descriptions, nous la calculons à partir de la mesure S telle que :

$$SPD(e_{\mathcal{C}}, e_{\mathcal{C}'}) = \frac{S(e_{\mathcal{C}}, e_{\mathcal{C}'})}{S(e_{\mathcal{C}}, e_{\mathcal{C}})} \tag{5.11}$$

En divisant $S(e_{\mathcal{C}}, e_{\mathcal{C}'})$ par $S(e_{\mathcal{C}}, e_{\mathcal{C}})$, on obtiendra toujours une SPD de 1 en comparant une description à elle-même. Étant donné une partition, on attend que 1 soit la plus petite des valeurs obtenues en comparant deux à deux les descriptions des clusters de la partition. Trouver des valeurs plus grandes est attendu. À l'inverse, trouver des valeurs inférieures est un indicateur, à la manière de l'IR, que la tache de description a rencontré une difficulté.

$SPD(e_{\mathcal{C}}, e_{\mathcal{C}'}) < 1$ indique que la distance entre les mots de $e_{\mathcal{C}}$ est plus grande que celle avec les mots de $e_{\mathcal{C}'}$. Comme nous l'avons expliqué, un tel cas de figure pourrait indiquer que les tags de $e_{\mathcal{C}}$ sont issus de contextes où ils apparaissent moins souvent que les tags de $e_{\mathcal{C}'}$.

5.3 Résultats, discussions et conclusions

Nous avons mené les expériences pour vérifier nos hypothèses de travail et pour obtenir une première chaîne de traitement à même de traiter des jeux de données classiques.

5.3.1 Résultats et analyses

Même en utilisant l'initialisation K-Means++, K-Means est un algorithme semi-aléatoire. Il faut donc s'assurer que les résultats sur lesquels nous basons nos analyses sont robustes et ne sont pas des échantillons de résultats aléatoires sélectionnés arbitrairement. Pour ce faire, nous avons répété le clustering plusieurs centaines de fois sur les mêmes données. Nous constatons que les partitions obtenues sont composées de clusters dont les tailles varient peu d'un clustering à l'autre. Comme la numérotation des clusters change d'une exécution à l'autre, nous reconnaissons les clusters aux identifiants des tweets qu'ils contiennent. Ainsi, nous pouvons constater qu'un cluster est identique à son homologue dans une autre partition, à entre 10 et 50 tweets près pour des clusters comptant entre 700 et 4700 tweets. De plus, la somme finale du score des descriptions obtenues varie en moyenne de 0,08% sur toutes les exécutions que nous avons réalisées.

Les résultats présentés ici sont ceux pour une exécution du clustering, sélectionnée aléatoirement. Nous donnons les résultats pour les méthodes de sélection de candidats descripteurs présentées en section 5.1.2. Le paramètre α de la contrainte de l'équation 5.8 peut être définit individuellement pour chaque cluster. Le maximum commun atteint pour ce paramètre pour les trois méthodes étant 0,3, nous utilisons cette valeur pour établir une base de comparaison.

Les clusters et leur description sont alignés dans les tableaux présentant les résultats. Le meilleur score sur une ligne du tableau est écrit en orange. Plus la valeur d'une cellule du tableau est bonne sur sa ligne, plus sa couleur est foncée. Ainsi, il est facile d'estimer si les descriptions obtenues sont bonnes au regard de nos critères en regardant le tableau correspondant. Celui-ci serait, pour le meilleur ensemble de description possible, bleu foncé uniquement sur sa diagonale d'en haut à gauche à en bas à droite, avec les valeurs en oranges uniquement sur cette diagonale. Les autres cellules seraient blanches ou peu colorées. Les descriptions pour chaque sélection de candidats sont effectuées à partir du même clustering pour être comparable.

Toutes les expérimentations sont menées sur un ordinateur portable, doté d'un i7-11850H et de 32Go de RAM. Le temps d'exécution total est inférieur à deux minutes pour la sélection par haute fréquence et à base de TF-IDF. Cela prend huit minutes de plus avec la sélection hybride, du fait de la recherche avec Farthest Point First.

5.3.1.1 Sélection par fréquence

La Figure 5.5 présente l'évaluation des descriptions obtenues avec des descripteurs identifiés sur la base de leur fréquence. Sur le tableau 5.5a, on peut constater que les descriptions obtiennent le score le plus élevé pour leur cluster dans 7 cas sur 10. La majorité des cellules du tableau ne sont pas colorées. Le seul cas problématique sur cet aspect est celui de la description 8. L'IR indique donc que les descriptions construites à partir de tags identifiés selon leur fréquence seulement semblent permettre de distinguer leur cluster des autres.

Le tableau 5.5b présentant la SPD. Il n'y a qu'un cas sur 10 où 1.0 n'est pas la plus petite valeur de la ligne, soit où les mots de la description sont moins proches entre eux que de ceux d'une autre description. Dans le tableau 5.5c, on peut voir que ce cas correspond à une description courte, composée de mots ambigus. La matrice de SPD apparaît assez peu bleutée. Elle permet donc d'observer le même comportement que celle d'IR, d'une manière moins marquée toutefois.

Finalement, le tableau 5.5c met en évidence le problème de cette sélection de candidats. Sur les 200 potentiels candidats que chaque sélection pourrait identifier, celle par la fréquence seule n'en propose que 84 dont 38 uniques. Cette faible diversité explique pourquoi certaines descriptions sont courtes. Ce phénomène peut s'expliquer de deux façons. D'une part, le vocabulaire employé pour parler de la tempête pourrait être assez peu diversifié. Ainsi, seuls les mots spécifiques à certains de ses aspects auraient des distributions nettement différentes d'un cluster à l'autre. D'autre part, le plongement pourrait avoir du mal à séparer les mots dans son espace, et par extension les documents. Cela pourrait tout aussi bien venir du manque de diversité dans le vocabulaire que nous venons d'évoquer que d'un paramétrage inadéquat de l'entraînement du Doc2Vec. Ce dernier cas de figure est exploré plus avant dans le chapitre 6.

À partir du même clustering toutefois, les autres sélections de candidats détectent une diversité de descripteurs plus importante comme nous allons le voir. Nous ne la retenons pas comme une méthode de sélection robuste.

Notons cependant que le contenu des descriptions, dans le tableau 5.5c, et leur évaluation permettent de mettre en évidence certains aspects des données. Par exemple, le cluster 3 a une très bonne évaluation en IR et en SPD. De plus sa description ne comporte que des mots qui ne sont pas apparus en doublons dans la liste initiale des candidats. Ceux-ci font, dans ce jeu de données, partie d'une comptine qu'un utilisateur de Twitter a re-publié environ 200 fois au cours de la tempête, en en changeant certains éléments. Il apparaît que cet aspect très particulier des données a été isolé. D'une part, ses mots sont au pire quinze fois plus distinctifs de ce cluster que des autres, ce que l'on peut voir dans le tableau 5.5a. D'autre part, ils sont au pire presque trois fois plus proches entre eux que des mots d'une autre description, comme on le voit dans le tableau 5.5b. Ainsi, ces comptines sont clairement distinguables au sein des données.

Une observation très similaire peut être effectuée pour le cluster numéroté 7. La violence et la vitesse des vents dans le Morbihan ressortent également comme des éléments

bien identifiés des autres dans nos données. Si le vent est un élément attendu d'une tempête, on peut noter qu'ici il est tout particulièrement associé à un département en particulier et à sa vitesse. Il se trouve que les vents les plus violents pendant Alex ont été mesurés à Belle-Île-en-Mer.

Il semblerait donc que les évaluations par IR et SPD permettent de mettre en évidence les éléments caractéristiques des données que les descriptions contiennent.

Réciproquement, les descriptions qui obtiennent de mauvaises évaluations avec nos critères contiennent des termes plus génériques ou ambigus. C'est le cas pour les 4, 5 et 8 ici, où les lignes sont colorées et/ou la meilleure valeur n'est pas toujours sur la diagonale. Les 4 et 5 ne contiennent que des mots dont on ne peut pas tirer de sens dans ces données sans plus de contexte. Dans la 8, 'image' et 'dégât' permettent de tirer du sens de la description. Il apparaît toutefois dans l'évaluation en IR que ces mots ne permettent pas d'isoler leur cluster des autres.

Le constat est moins bon pour les données sur la guerre Russo-Ukrainienne. Les deux tableaux d'évaluations 5.6a et 5.6b sont beaucoup plus colorés. Dans le premier, la meilleure valeur des lignes n'est sur la diagonale que 4 fois sur 10. La matrice de SPD apparaît fortement colorée, ce qui signifie que les mots des différentes descriptions sont assez peu séparés sémantiquement les uns des autres.

Nous pouvons voir en outre un cluster "cas particulier" à la manière de celui de comptine pour Alex, ici le 4. Celui-ci obtient d'excellentes évaluations mais cela vient du fait que le cluster ne compte que 21 tweets sur les 18311 du corpus. Ces derniers sont systématiquement identifiés comme un cluster à part entière par K-Means. La ligne 6 du tableau 5.6b explique ce phénomène. Les mots des messages concernés sont en effet au moins 95 fois plus proches entre eux que des autres mots du corpus. En étudiant manuellement le contenu de ce cluster, nous ne parvenons pas à lui trouver un sens global. En revanche, presque tous les messages contiennent des adresses (des tokens formés d'une '@' et du nom du twittos) qui n'apparaissent nulle part ailleurs dans les données. Nous pouvons voir dans le tableau 5.6a qu'aucun des mots de la description 4 n'apparaît dans d'autres clusters, puisque son IR est de 0 pour tout autre cluster que le 4.

Dans ce corpus, nous pouvons constater dans le tableau 5.6b qu'encore moins de candidats sont identifiés par leur fréquence que pour le corpus sur la tempête, avec seulement 79 tags utilisables par le modèle. Il semble que le vocabulaire employé pour parler de la guerre soit moins diversifié que celui pour aborder une tempête.

Les résultats de la sélection par les plus hautes fréquences sur les données sur la guerre en Ukraine nous encouragent également à ne pas retenir cette sélection de candidats.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	3.211	0.1853	0.695	0.4994	0.6059	0.2593	2.2343	0.6492	0.5956	0.7037
1	0.2553	3.3376	1.2811	1.1396	1.4708	1.6965	0.3492	0.3448	0.4685	0.2501
2	0.3348	0.9078	1.1652	0.5562	0.9996	0.496	0.2928	1.1293	0.2454	0.5408
3	0.118	0.0944	0.147	2.1884	0.0991	0.0939	0.173	0.0725	0.4068	0.0736
4	0.5126	1.2268	0.7053	0.6521	1.0308	0.7163	0.3657	0.9552	0.3359	0.6215
5	0.1211	1.3471	0.3485	0.4815	0.5254	1.1437	0.1204	0.3916	0.3878	0.407
6	3.1251	0.1356	0.2583	0.2882	0.3625	0.1957	4.3659	0.2945	0.8339	0.7921
7	0.3447	0.7896	0.7444	0.5337	0.8001	0.6221	0.2649	6.6448	0.6248	3.2539
8	1.1588	0.9869	0.707	0.9099	0.83	0.9844	1.3012	1.0867	0.8983	1.1104
9	1.0506	0.3766	0.3611	0.4598	0.5793	0.486	0.8464	3.4863	0.9011	4.7613

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	2.105	1.8721	1.7084	1.82	2.1747	1.0333	1.963	1.7237	1.6448
1	5.5589	1.0	2.6067	4.5345	2.526	1.9304	5.3447	4.8898	4.2964	5.2869
2	2.0941	1.1042	1.0	1.6717	1.1922	1.095	1.946	1.5506	1.6085	1.7497
3	3.1291	3.1451	2.7372	1.0	3.2407	3.2724	3.4179	3.4573	3.6095	3.4451
4	1.8515	0.9731	1.0842	1.7999	1.0	1.0504	1.7226	1.5484	1.5448	1.6123
5	3.5269	1.1855	1.5875	2.8974	1.6745	1.0	3.4307	2.5522	2.4302	2.5485
6	1.4257	2.7924	2.4002	2.5746	2.3363	2.9187	1.0	2.6197	2.262	2.102
7	2.7537	2.5975	1.9445	2.6479	2.1352	2.2077	2.6636	1.0	2.3829	1.601
8	1.3506	1.2747	1.1266	1.544	1.1898	1.1741	1.2845	1.3309	1.0	1.3014
9	2.4997	3.0426	2.3772	2.8586	2.4086	2.3882	2.3153	1.7345	2.5244	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'#alpesmaritime', 'sinistre', 'pompier', 'deux', 'direct', 'village', '#nice06', 'vallee', 'habitant', 'toujour'
1	'c'est', 'faire', 'bien', 'quand', 'meme', 'j'ai', 'dire', 'pouvoir', 'trop', 'voir'
2	'temp', 'comme', 'matchedNumber', 'grand', 'nouveau', 'dernier', 'cette'
3	'dune', 'ainsi', 'bout', 'trois', 'bois', 'vivement', 'dansaient', 'capucine', 'virant'
4	'plus', 'pluie', 'jour', 'fait', 'aussi', 'merci'
5	'aller', 'cest', 'chez', 'demain', 'calme'
6	'alpes-maritime', 'personne', 'disparu', 'passage', 'corps', 'mort', 'franceinfo', 'retrouve', 'crue', 'bilan'
7	'vent', 'nuit', 'rafale', 'kilometre', '#bretagne', 'fort', '#meteo', '#morbihan', 'violent', 'cote'
8	'tout', 'apre', 'degat', 'avant', 'cause', 'vendredi', 'image'
9	'france', 'bretagne', 'meteo', 'morbihan', 'vigilance', 'jeudi', 'rouge', 'alerte', 'orange', 'departement'

(c) Description des clusters

FIGURE 5.5 – Évaluation des descriptions sur les données sur la tempête Alex, pour la sélection par fréquence ayant 84 candidats descripteurs, dont 38, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	1.4036	0.0339	2.051	0.1335	0.0	0.2698	1.2133	0.0304	0.0778	0.2273
1	0.5183	0.8651	0.4555	0.9083	0.1374	0.507	0.8465	1.2171	0.5409	0.4737
2	1.3484	0.2911	8.0933	0.2563	0.0	0.585	0.4583	0.1358	0.2034	0.5286
3	0.3971	0.6172	0.3509	0.9491	0.5464	0.4383	0.4101	1.1686	0.6057	0.3056
4	0.0	0.0	0.0	0.0	38.8528	0.0	0.0	0.0	0.0	0.0
5	0.8096	0.3528	1.2327	0.3659	0.4995	0.8903	1.2638	0.3153	0.565	0.526
6	2.274	0.7297	2.657	0.7316	0.1073	1.0665	4.164	0.8025	0.6186	0.8466
7	0.7536	1.4557	0.4679	2.0545	0.3108	0.8313	1.1123	3.0389	1.0033	0.6479
8	0.5266	0.5198	0.5751	1.0254	0.7695	0.6996	0.5936	1.0517	1.0201	0.4474
9	0.7889	0.7709	1.4056	0.6419	0.1574	0.8306	1.0142	0.7752	0.5502	0.75

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	2.517	1.4237	2.535	0.9447	1.4062	1.5232	2.614	2.4759	1.9164
1	3.8297	1.0	3.7618	1.3743	2.8889	3.0119	2.5723	1.2727	2.6274	1.8481
2	1.9226	3.3389	1.0	3.4414	1.1604	2.41	2.9836	3.8362	3.0018	2.4516
3	4.8054	1.7123	4.8307	1.0	2.8945	4.8237	3.7914	1.0563	2.7618	3.189
4	104.7508	210.5365	95.2824	169.3111	1.0	176.6747	184.0697	187.6189	172.8528	182.4683
5	1.8113	2.5499	2.2987	3.2777	2.0523	1.0	1.714	3.2854	3.3348	1.7279
6	1.4003	1.5543	2.0311	1.8388	1.5261	1.2234	1.0	1.7382	2.3235	1.3756
7	5.9672	1.9094	6.4844	1.272	3.8625	5.8225	4.3159	1.0	3.5748	3.9326
8	1.4371	1.0024	1.2903	0.8457	0.9049	1.5028	1.467	0.909	1.0	1.2039
9	1.8081	1.146	1.7128	1.5872	1.5526	1.2656	1.4117	1.6254	1.9568	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'civilian', 'humanitarian', 'provide', 'death', 'assistance', 'europarl_en'
1	'putin', 'world', 'want', 'invade'
2	'potus', '#stopputin', 'nato', 'close', 'security', '#stoprussia', 'council', 'exclude', 'vonderleyen', 'eucopresident'
3	'like', 'make', 'take', 'biden'
4	'chuck1one', 'edmemphispimpin', 'trace_avp', 'selldateamdolan', 'yankeesknicks99', 'not47buddz', 'legionofknick', 'freekeith', 'stalling_e', 'mikefromqns7'
5	'russian', 'ukrainian', '#russia', 'force'
6	'stop', 'need', 'people', 'weapon', 'defend', '#putin', 'innocent', 'help', 'kill', 'please'
7	'country', 'would', 'think', 'time', 'know', 'right', 'trump', 'good', 'well', 'thing'
8	'matchedNumber', 'military', 'year', 'even', 'give', 'state'
9	'russia', 'invasion', 'support', 'fight', 'aggression'

(c) Description des clusters

FIGURE 5.6 – Évaluation des descriptions sur les données sur la guerre en Ukraine, pour la sélection par fréquence ayant 79 candidats descripteurs, dont 42, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

5.3.1.2 Sélection des tops DF-IDF

Les tableaux de la figure 5.7 montrent l'évaluation des descriptions obtenues avec les candidats descripteurs ayant les meilleurs scores DF-IDF. On peut y constater une certaine amélioration de l'évaluation d'IR par rapport aux descripteurs identifiés par leur fréquence. La ligne 8 du tableau 5.7a n'est plus uniformément colorée, même si sa meilleure valeur n'est toujours pas sur sa diagonale. Cela reste d'ailleurs le cas pour 3 cas sur 10. Ainsi, les descriptions déjà bien identifiées précédemment le sont toujours et un des cas problématique l'est un peu moins ici. Nous pouvons également noter que la ligne 4 est toujours assez colorée. Le meilleur score de cette description est obtenu pour le cluster 1. C'est pour ce cluster que les descriptions 4 et 5 obtiennent leur meilleure IR, pour les candidats par fréquence comme ceux par les tops DF-IDF. Dans les tableaux présentant le contenu des descriptions, on peut voir que les 1, 4 et 5 sont majoritairement composées de mots que l'on peut interpréter de différentes façons. Il semble donc que le manque de spécificité dans les unités composant les messages du corpus peuvent compliquer la séparation des cluster et/ou la description de la partition.

Notons également que le fait que la ligne 8 soit bien moins colorée dans le tableau 5.7a que dans le 5.5a correspond au fait que cette description compte plus d'éléments avec les candidats DF-IDF qu'avec ceux par fréquence. En particulier, on peut noter l'apparition de `saint-martin-vesubie`, d'`intemperie`, d'`image` et de `video`. La tempête a effectivement causé des dégâts particulièrement importants dans Saint-Martin-Vésubie, une commune des Alpes-Maritimes. La description 8 obtient de plus sa meilleure IR pour le cluster 6, où le nom de ce département apparaît comme descripteur. Dans les données, environ une centaine de tweets mentionnent Saint-Martin-Vésubie alors que plusieurs milliers parles des effets de la tempête dans les Alpes-Martimes. Cette observation semble indiquer que des éléments de petite granularité sont bel et bien détectés dans les données, mais que leur lien sémantique avec un élément englobant peut poser problème à un score de saillance comme le DF-IDF.

Le tableau 5.7b est plus coloré que le 5.5b. La meilleure valeur de ligne est sur la diagonale toujours dans 9 cas sur 10. Le cas problématique est pour la SPD aussi lié au clusters 1 et 5, ce qui rejoint l'observation établie avec l'évaluation d'IR. On y voit les mots des descriptions 1 et 6 assez proches, ce que l'on constate aussi dans le tableau 5.5c. Ces deux clusters y semblent concerner des évènements qui ont eu lieu dans le même département. Nous retrouvons également les clusters 3 et 7 aussi bien distingués que précédemment. Les descriptions 1, 4 et 5 sont par ailleurs toujours difficilement distinguables, et le tableau 5.5c montre qu'elles contiennent toujours des mots ambigus.

La figure 5.8 montre les mêmes tendances que les descriptions des données sur la guerre en Ukraines basées sur les fréquences. Les cas où la meilleure valeur de la ligne est ou n'est pas sur la diagonale sont les mêmes, et la coloration des deux tableaux d'évaluation est très similaire. La description 2 obtient une bonne évaluation pour les deux sélections. Elle est en fait identique dans les deux cas.

De manière générale, les deux ensembles de description sont similaires. Les 13 descripteurs supplémentaires par rapport à ceux que la fréquence a identifiés sont dans les descriptions 3, 5 et 8. Toutefois, leur apport à la compréhension des données est limité. En effet, on peut constater dans le tableau 5.8c dans la description 8 l'apparition de différents noms d'entreprise de production ou de diffusion de contenu audiovisuel (`netflix`, `disney`, `wetv`, `hotstar`) sans rien pour comprendre pourquoi ils sont mentionnés. Un nouveau mot

intéressant dans la description 5 est `attack`, mais il n'améliore pas vraiment son évaluation. On peut toutefois noter que sa présence contribue à faire obtenir à la description 5 sa meilleure IR pour le cluster 2, plutôt que pour le 6 comme dans le tableau 5.6b.

De plus, la description 5 obtient une IR plus élevée pour le cluster 6, comme c'est le cas avec la sélection par fréquence. La 2 fait référence à la réaction internationale à la guerre en Ukraine. Le 5 semble faire référence à l'offensive Russe. La 6 quant à elle parle de la protection des civils ukrainiens. L'évaluation en IR semble donc indiquer que ces trois notions sont liées dans ce corpus, comme cela est attendu compte-tenu de sa cible.

L'évaluation de la SPD des descriptions dans le tableau 5.8b montre 8 cas sur 10 où la meilleure valeur de la ligne est sur la diagonale. La coloration générale du tableau tend à indiquer que les descriptions ne sont pas clairement distinguées au regard de la sémantique capturée par le modèle de plongement. Celle des lignes 1, 3 et 7 montre que les descriptions correspondantes sont relativement proches. Dans leur contenu, visible dans le tableau 5.8c, nous pouvons voir effectivement qu'elles contiennent `putin, biden` et `trump`, et qu'elles semblent être à propos de leurs échanges.

Une observation surprenante est la coloration de la colonne 4 du tableau 5.8b. Elle est relativement uniforme, indiquant que plusieurs descriptions comportent des mots proches de ceux de la description 4. Elle comporte même deux valeurs en orange de plus que celle qu'on attend pour la ligne 4. Dans le même temps, les faibles valeurs d'IR sur la colonne 4 indique que les mots des autres descriptions n'apparaissent pas ou peu dans les 21 tweets du cluster 4.

Cela s'explique par la construction de l'espace de plongement. Les vecteurs de documents du cluster 4 sont extrêmement proches entre eux, beaucoup plus que d'aucun autre document, expliquant l'identification de ce cluster. Les mots de ces documents n'apparaissent quasiment que dans ces documents. Ainsi, comme Doc2Vec construit ses représentations en mettant à jour des vecteurs de zéros, ces mots sont toujours proches du point neutre de l'espace de plongement. Ils n'ont en effet été rencontrés dans aucun autre contexte, et n'ont donc été rapprochés d'aucun autres mots. Comme les mots des autres descriptions sont plus dispersés dans l'espace, la distance entre eux peut être plus élevée qu'avec ceux de la description 4.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	3.2788	0.1086	0.5426	0.4021	0.5159	0.1795	2.3594	0.5817	0.526	0.6237
1	0.2271	3.3145	1.1892	1.025	1.3074	1.6839	0.3111	0.2816	0.4232	0.1987
2	0.644	0.8267	1.356	0.5388	0.8626	0.4059	0.5441	0.9502	0.294	0.4187
3	0.3359	0.4536	0.4109	2.5352	0.3854	0.3383	0.3246	0.1761	0.5794	0.1908
4	0.6106	1.6855	1.291	0.9614	1.5849	0.9415	0.4535	1.2075	0.4418	0.5434
5	0.2437	2.0161	0.5937	0.7958	0.9233	1.7661	0.3102	1.0976	0.6966	1.0289
6	2.8064	0.1336	0.2396	0.2648	0.3191	0.2026	4.2494	0.2558	0.7907	0.658
7	0.3038	0.8288	0.8664	0.576	0.9009	0.672	0.2117	6.8268	0.6401	3.3958
8	2.0807	0.4952	0.4537	0.6104	0.6388	0.557	2.3844	1.3851	1.035	1.7571
9	0.889	0.3179	0.3101	0.3948	0.5088	0.4235	0.5693	3.4149	0.8333	4.638

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	2.5726	1.8355	1.9814	2.1659	2.5512	1.1008	2.3453	1.5084	1.9028
1	6.1754	1.0	3.3817	4.6483	2.3267	2.6046	5.5384	5.0777	5.3886	5.622
2	1.3913	1.0678	1.0	1.2841	1.0395	1.143	1.3396	1.2796	1.2701	1.322
3	2.5571	2.4991	2.1864	1.0	2.2225	2.7638	2.7501	2.9012	3.162	2.8248
4	2.0568	0.9205	1.3024	1.6354	1.0	1.2228	1.9092	1.6513	1.8844	1.848
5	2.3667	1.0066	1.3989	1.9866	1.1945	1.0	2.2231	1.5894	1.8596	1.5948
6	1.3481	2.8257	2.1644	2.6097	2.4622	2.9349	1.0	2.8417	1.7262	2.3389
7	2.8959	2.612	2.0846	2.7758	2.1472	2.1157	2.8651	1.0	2.2701	1.5951
8	1.3048	1.9417	1.4494	2.1193	1.7164	1.7339	1.2192	1.5902	1.0	1.4063
9	2.7705	3.4101	2.5396	3.187	2.8335	2.5032	2.7807	1.8809	2.3673	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'bilan', 'pompier', '#alpesmaritime', 'sinistre', 'deux', 'direct', 'village', '#nice06', 'habitant', 'secours'
1	'c'est', 'faire', 'bien', 'quand', 'meme', 'j'ai', 'dire', 'trop', 'dehors', 'beau'
2	'temp', 'voir', 'dernier', 'nouveau', 'cette', '#france', 'lequel', 'verite', 'cote', 'vallee'
3	'tout', 'dune', 'bout', 'dansaient', 'capucine', 'virant', 'bois', 'vivement', 'ainsi', 'trois'
4	'plus', 'pouvoir', 'comme', 'fait', 'jour', 'merci', 'aussi', 'grand', 'd'une', 'terre'
5	'aller', 'cest', 'chez', 'demain', 'avant', 'calme', 'cause', 'arriver', 'pluie'
6	'alpes-maritime', 'disparu', 'personne', 'passage', 'corps', 'mort', 'crue', 'retrouve', 'emporte', 'italie'
7	'vent', 'matchedNumber', 'rafale', 'kilometre', '#bretagne', 'nuit', '#meteo', '#morbihan', 'fort', 'violent'
8	'apre', 'franceinfo', 'france', 'saint-martin-vesubie', 'degat', 'vendredi', 'image', 'premier', 'intemperie', 'video'
9	'bretagne', 'meteo', 'morbihan', 'jeudi', 'vigilance', 'rouge', 'alerte', 'orange', 'departement', 'l'ouest'

(c) Description des clusters

FIGURE 5.7 – Évaluation des descriptions sur les données sur la tempête Alex, pour la sélection des Tops DF-IDF ayant 99 candidats descripteurs, dont 50, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	1.7335	0.0685	2.6991	0.1829	0.0	0.417	1.6338	0.0767	0.1283	0.3033
1	0.5015	0.8047	0.3932	0.8416	0.1374	0.4958	0.8084	1.1147	0.5769	0.4488
2	1.3484	0.2911	8.0933	0.2563	0.0	0.585	0.4583	0.1358	0.2034	0.5286
3	0.7011	1.0125	0.5286	1.719	0.6633	0.7502	0.7744	2.096	1.1127	0.586
4	0.0	0.0	0.0	0.0	38.8528	0.0	0.0	0.0	0.0	0.0
5	0.9234	0.3891	1.6516	0.4468	0.3598	1.0334	1.4055	0.3516	0.7182	0.6026
6	2.1893	0.7941	2.7951	0.7906	0.247	1.1751	4.1214	0.8699	0.7172	0.8929
7	0.7716	1.5578	0.5383	2.1277	0.5466	0.867	1.101	3.1205	1.0684	0.6783
8	0.134	0.1232	0.1608	0.2962	2.5698	0.5421	0.0583	0.2154	1.8563	0.1247
9	0.7889	0.7709	1.4056	0.6419	0.1574	0.8306	1.0142	0.7752	0.5502	0.75

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	2.08	1.2247	2.159	0.8786	1.2207	1.3877	2.2968	1.0814	1.6466
1	3.3032	1.0	3.5308	1.0775	2.2943	3.104	2.2717	1.0186	2.8461	1.9969
2	1.9038	3.4562	1.0	3.3317	1.1604	2.0222	2.9673	3.7141	1.2327	2.4516
3	4.2563	1.3377	4.2253	1.0	2.4515	4.2719	3.3986	1.0899	3.0454	2.8274
4	112.1433	184.4049	95.2824	158.7191	1.0	163.2618	182.4811	192.8649	25.0675	182.4683
5	1.4768	2.3647	1.5738	2.6216	1.5475	1.0	1.575	2.7329	1.7622	1.4436
6	1.769	1.8235	2.4333	2.1976	1.8225	1.6595	1.0	2.1346	2.2529	1.568
7	4.3687	1.22	4.5445	1.0516	2.8741	4.2966	3.185	1.0	3.5133	2.7596
8	3.7028	6.1365	2.7151	5.2893	0.6725	4.9873	6.0513	6.3246	1.0	5.7594
9	1.7882	1.3657	1.7128	1.5576	1.5526	1.2959	1.3358	1.5757	1.8268	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'civilian', 'weapon', 'humanitarian', 'provide', 'death', 'assistance', 'europarl_en', 'emmanuelmacron'
1	'putin', 'world', 'want', 'first'
2	'potus', '#stopputin', 'nato', 'close', 'security', '#stoprussia', 'council', 'exclude', 'vonderleyen', 'eucopresident'
3	'call', 'make', 'biden', 'right', 'take', 'back', 'well', 'even', 'year'
4	'chuck1one', 'edmemphispimpin', 'trace_avp', 'selldateamdolan', 'yankeesknicks99', 'not47buddz', 'legionofknick', 'freekeith', 'stalling_e', 'mikefromqns7'
5	'russian', '#russia', 'attack', 'military', 'force', 'borisjohnson'
6	'stop', 'need', 'defend', '#putin', 'innocent', 'people', 'ukrainian', 'help', 'kill', 'please'
7	'country', 'would', 'like', 'think', 'time', 'trump', 'know', 'good', 'invade', 'thing'
8	'matchedNumber', 'canva', 'wetv', 'vidio', 'netflix', 'disney', '#tigray', 'spotify', 'jual', 'hotstar'
9	'russia', 'invasion', 'support', 'fight', 'aggression'

(c) Description des clusters

FIGURE 5.8 – Évaluation des descriptions sur les données sur la guerre en Ukraine, pour la sélection des tops DF-IDF ayant 92 candidats descripteurs, dont 52, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

5.3.1.3 Sélection hybride

L'incorporation FPF de la sélection hybride, décrite en section 5.1.2.3, lui permet de sélectionner des candidats plus couvrants. FPF est en effet un alogrtihme cherchant des points les plus éloignés possibles dans l'espace exploré.

L'objectif de la sélection hybride est d'identifier des descripteurs répartis plus largement sur l'espace de représentation des mots qui apparaissent dans un cluster. Elle identifie ainsi une plus grande variété de candidats. Pour garantir la pertinence d'un tag, nous imposons un score DF-IDF minimum pour qu'il puisse être sélectionné comme candidat. Ainsi le tableau 5.9a présentant l'IR des descriptions avec la sélection hybride montre les mêmes tendances que le tableau 5.7a des candidats DF-IDF. Nous pouvons voir que la meilleure valeur de la ligne 4 apparaît maintenant sur sa diagonale. Celle de la ligne 5 reste sur la colonne du cluster 1. De manière générale, le tableau 5.9a est plus coloré que le 5.7a. Cette observation est attendue, comme les mots ne sont pas choisis uniquement sur leur propension à distinguer leurs clusters. Il semble ainsi que la sélection hybride n'impacte pas significativement les descriptions obtenues du point de vue de leur IR.

Le changement est visible avec la SPD et le contenu des descriptions. Le tableau 5.9b n'est presque pas coloré, ce qui signifie que les mots des descriptions sont dans la majeure partie des cas aussi proches entre eux que des mots d'une autre description. C'est une conséquence attendue de la sélection hybride, puisqu'elle choisit des mots plus dispersés dans l'espace de représentation. Nous pouvons voir que la colonne 1 contient 5 valeurs en orange. Dans le tableau 5.9c, nous pouvons voir que la description 1 ne comporte presque que des mots auxquels on ne peut pas attribuer de sens seul. Il semblerait donc que ce type de mot soit disposé à des distances similaires des autres au sein de l'espace de représentation. Cela s'explique du fait qu'ils apparaissent dans différents contextes. En caricaturant, on peut dire que le modèle de plongement les place à mi-chemin entre les zones sémantiques de l'espace associé à ces contextes.

En terme de contenu, les descriptions restent similaires. Nous pouvons constater dans le tableau 5.9c que le sens qu'il est possible de tirer des descriptions reste proche de celui ressortant du tableau 5.7c. On peut noter l'apparition du tag `catastrophe` dans la description 4, mais les autres restent difficiles à interpréter sans contexte. La description 7 parle toujours de vent, mais on n'y trouve plus d'élément le liant au record de vitesse établit en Bretagne. Notons également que `saint-martin-vesubie` est passé de la description 8, où il était associé à `degat`, à la 6, où l'on retrouve `nicois`, `personne`, `disparu` et `crue`. Les associations de mots ne sont plus les mêmes, mais on retrouve là les évènements que nous avons évoqués dans les résultats des autres sélections.

Il apparaît donc que si le clustering permet bel et bien mettre en évidence les sous-évènements que nous recherchons, les frontières sémantiques entre les clusters ne sont pas imperméables.

La même tendance d'observation peut être faite dans le tableau 5.10 pour l'évaluation en IR des données sur la guerre en Ukraine. Nous pouvons voir dans leur tableau de SPD, le 5.10b, que les valeurs de la ligne 4 sont ici du même ordre de grandeur que celles des autres lignes. La description 4 obtient toujours sa meilleure IR pour son cluster. Comme la sélection hybride offre plus de candidats, on peut voir dans le tableau 5.10c que les descriptions sont plus longues. Par exemple, les tags `troop`, `conventional` et `protest` apparaissent dans la description 9 d'où est disparu `agression`. Le sens de cette description en est devenu plus difficile à discerner.

Cette constatation pointe vers le fait qu'avoir des descriptions plus longues n'est pas nécessairement une bonne chose. La variation du sens que l'on peut tirer des descriptions nous indique par ailleurs que les clusters peuvent contenir des informations sur différents objets.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	2.6248	0.1279	0.5207	0.3446	0.576	0.1998	1.7913	0.2392	0.4428	0.319
1	0.1532	1.6093	0.5964	0.5549	0.7327	0.6451	0.1675	0.2148	0.2532	0.1632
2	0.7572	1.1773	1.2325	0.6851	1.0442	0.5847	0.665	0.8132	0.362	0.485
3	0.7222	0.6832	0.4665	1.277	0.4912	0.4352	0.6515	0.3582	0.4693	0.2565
4	0.9537	1.0065	0.7768	0.6696	1.1138	0.5707	0.7687	0.8142	0.4017	0.4874
5	0.2016	1.2651	0.4901	0.5457	0.6228	1.1936	0.2999	0.4828	0.5524	0.5634
6	2.1585	0.0917	0.1443	0.2457	0.2417	0.139	3.0231	0.1207	0.5845	0.4012
7	0.5851	0.9036	0.7629	0.5636	0.9703	0.6256	0.4348	4.1659	0.5613	2.0982
8	1.9032	0.4289	0.4768	0.6042	0.6392	0.595	2.0747	0.9206	1.0399	1.2013
9	0.7392	0.2504	0.2864	0.3458	0.3854	0.3118	0.67	1.7323	0.5475	2.7475

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	2.3527	2.0123	1.8135	1.7063	2.4237	1.1078	2.3378	1.8354	1.9027
1	4.5198	1.0	2.5549	3.1863	2.9487	2.0983	4.4572	3.7324	4.1953	3.8611
2	1.2553	0.8296	1.0	1.155	1.061	1.0401	1.2094	1.2178	1.3397	1.196
3	1.0805	0.9882	1.1031	1.0	1.0678	1.1218	1.0893	1.2335	1.2087	1.1397
4	1.1104	0.9989	1.1069	1.1663	1.0	1.1435	1.0439	1.3189	1.2745	1.1167
5	1.8714	0.8433	1.2873	1.4538	1.3567	1.0	1.834	1.4038	1.6073	1.3982
6	1.4626	3.0632	2.5596	2.4138	2.1179	3.1361	1.0	2.5923	2.1861	2.1261
7	2.1649	1.7992	1.8078	1.9173	1.8768	1.6837	1.8183	1.0	1.7566	1.3127
8	1.1827	1.4071	1.3837	1.3073	1.2619	1.3413	1.0669	1.2223	1.0	1.1078
9	1.9058	2.0131	1.9203	1.916	1.7187	1.8138	1.613	1.4198	1.722	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'#alpesmaritime', 'mobilise', 'solidarite', 'roya', 'moin', 'village', '#nice06', 'vesubie', 'pompier', 'soutien'
1	'donc', 'qu'il', 'vouloir', 'hasard', 'coup', 'parler', 'mieux', 'bonjour', 'juste', 'j'ai'
2	'sans', 'temp', 'd'un', 'pensee', '#france', 'tres', 'pouvoir', 'plein', 'mettre', 'nouveau'
3	'quand', 'covid', 'depuis', 'vivement', 'region', 'bout', 'maritime', 'ainsi', 'villefranche', 'sinistre'
4	'jour', 'situation', 'pendant', 'rester', 'catastrophe', 'minute', 'autre', 'monde', 'plus', 'aider'
5	'cause', 'avant', 'comment', 'calme', 'arriver', 'alpe', 'commencer', 'recolter', 'quoi', 'passer'
6	'saint-martin-vesubie', 'personne', 'portee', 'nice_matin', 'franceinfo', 'crue', 'nicois', 'disparu', 'retrouve', 'disparue'
7	'cote', 'pays', 'vent', 'ouestfrance', 'entre', 'kilometre', 'pluie', 'matin', 'octobre', 'heure'
8	'passage', 'loup', 'premier', 'macron', 'matchedHour', 'apre', 'degat', 'image', 'parc', 'ferme'
9	'rouge', 'ecole', 'inondation', 'meteo', 'bfmtv', 'frapper', 'etablissement', 'd'electricite', 'france', 'l'ouest'

(c) Description des clusters

FIGURE 5.9 – Évaluation des descriptions sur les données sur la tempête Alex, pour la sélection hybride ayant 112 candidats descripteurs, dont 73, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	1.1163	0.2416	1.2374	0.3317	0.0	0.5153	1.1788	0.2618	0.2676	0.363
1	0.6387	0.9338	0.5086	0.9299	0.1374	0.6884	0.9729	1.1994	0.5495	0.5546
2	1.7352	0.4284	6.2702	0.3719	0.0	0.7393	1.4072	0.2211	0.2929	0.7141
3	0.4763	0.654	0.4267	1.1692	0.0	0.5854	0.5732	1.2616	0.6851	0.4157
4	0.5884	0.6583	0.4798	0.9643	3.828	0.6675	0.6821	1.1519	0.8242	0.5011
5	0.7417	0.3629	1.0934	0.5063	0.0	0.9491	0.9092	0.3966	0.6164	0.5611
6	1.6153	0.7515	1.8787	0.6886	0.4383	1.0906	3.3199	0.8611	0.6475	0.82
7	0.5619	1.0363	0.4293	1.4574	0.1447	0.5953	0.7477	2.1996	0.7947	0.4835
8	0.5479	0.5246	0.6003	0.8539	0.7695	0.7416	0.5116	0.909	1.144	0.4405
9	0.7742	0.7431	0.9462	0.6451	0.0	0.8143	0.9155	0.7338	0.5185	0.7508

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	1.3839	1.393	1.3341	1.1173	1.0952	1.3291	1.6204	1.3619	1.1176
1	1.5139	1.0	2.2103	1.1003	1.046	1.7074	1.6104	1.2059	1.5037	1.2948
2	1.3434	1.9487	1.0	1.9169	1.8117	1.3976	1.7898	2.2574	1.7472	1.4171
3	1.9107	1.4406	2.8467	1.0	1.281	2.0853	2.6505	1.1955	1.5736	1.7219
4	1.4429	1.2348	2.4259	1.1551	1.0	1.6603	1.7742	1.3758	1.5455	1.3979
5	1.0234	1.4584	1.3542	1.3605	1.2014	1.0	1.4078	1.6285	1.3307	1.1763
6	1.3879	1.5373	1.9379	1.9325	1.4346	1.5733	1.0	2.1131	1.9847	1.3919
7	1.8668	1.27	2.6966	0.9616	1.2274	2.0078	2.3313	1.0	1.4818	1.633
8	1.0523	1.062	1.3997	0.8489	0.9246	1.1003	1.4684	0.9938	1.0	1.0387
9	1.1026	1.1676	1.4496	1.1861	1.0678	1.2419	1.315	1.3983	1.3263	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'weapon', 'hurt', 'osce_smm', 'civilian', 'death', 'russia'', 'count', 'leave', 'video', 'regular'
1	'news', 'country', 'stand', 'absolutely', 'join', 'want', 'world'
2	'aggression', 'security', 'nato', 'provide', 'zelenskyyua', '#stopputin', 'council', 'kyiv', 'potus', 'stop'
3	'president', 'biden', 'tran', 'next', 'long', 'face', 'watch', 'situation', 'china', 'even'
4	'like', 'many', 'learn', 'plan', 'force', 'care', 'killing', 'mean', 'today', 'live'
5	'meta', 'million', 'foreign', 'student', 'indian', 'russian', 'limit', 'jm_scindia', 'share', 'military'
6	'please', '#russia', 'defend', 'kill', 'people', 'send', 'soldier', 'fight', 'innocent', 'help'
7	'know', 'india', 'good', 'fool', 'nuclear', 'invade', 'year', 'conflict', 'trump', 'think'
8	'state', 'matchedNumber', 'vladimir', 'house', 'give', 'crime', 'find', 'direction', 'provocation', 'first'
9	'russia', 'invasion', 'troop', 'churchill', 'conventional', 'protest', 'support', 'safe'

(c) Description des clusters

FIGURE 5.10 – Évaluation des descriptions sur les données sur la guerre en Ukraine, pour la sélection hybride ayant 95 candidats descripteurs, dont 63, apparaissant en vert foncé, sont uniques dans l'ensemble initial des candidats.

5.3.1.4 Contenu des descriptions

Id	Contenu
0	'#alpesmaritime', 'pompier', 'village', '#nice06', 'sinistre', 'deux', 'direct', 'habitant', 'vallee', 'toujour', 'bilan', 'secours', 'mobilise', 'solidarite', 'roya', 'moin', 'vesubie', 'soutien'
1	'j'ai', 'c'est', 'faire', 'bien', 'quand', 'meme', 'dire', 'trop', 'pouvoir', 'voir', 'dehors', 'beau', 'donc', 'qu'il', 'vouloir', 'hasard', 'coup', 'parler', 'mieux', 'bonjour', 'juste'
2	'temp', 'nouveau', 'dernier', 'cette', '#france', 'comme', 'matchedNumber', 'grand', 'voir', 'lequel', 'verite', 'cote', 'vallee', 'sans', 'd'un', 'pensee', 'tres', 'pouvoir', 'plein', 'mettre'
3	'ainsi', 'bout', 'vivement', 'dune', 'trois', 'bois', 'dansaient', 'capucine', 'virant', 'tout', 'quand', 'covid', 'depuis', 'region', 'maritime', 'villefranche', 'sinistre'
4	'plus', 'jour', 'fait', 'aussi', 'merci', 'pluie', 'pouvoir', 'comme', 'grand', 'd'une', 'terre', 'situation', 'pendant', 'rester', 'catastrophe', 'minute', 'autre', 'monde', 'aider'
5	'calme', 'aller', 'cest', 'chez', 'demain', 'avant', 'cause', 'arriver', 'pluie', 'comment', 'alpe', 'commencer', 'recolter', 'quoi', 'passer'
6	'personne', 'disparu', 'retrouve', 'crue', 'alpes-maritime', 'passage', 'corps', 'mort', 'franceinfo', 'bilan', 'emporte', 'italie', 'saint-martin-vesubie', 'portee', 'nice_matin', 'nicois', 'disparue'
7	'vent', 'kilometre', 'nuit', 'rafale', '#bretagne', 'fort', '#meteo', '#morbihan', 'violent', 'cote', 'matchedNumber', 'pays', 'ouestfrance', 'entre', 'pluie', 'matin', 'octobre', 'heure'
8	'apre', 'degat', 'image', 'vendredi', 'premier', 'tout', 'avant', 'cause', 'franceinfo', 'france', 'saint-martin-vesubie', 'intemperie', 'video', 'passage', 'loup', 'macron', 'matchedHour', 'parc', 'ferme'
9	'meteo', 'rouge', 'france', 'bretagne', 'morbihan', 'vigilance', 'jeudi', 'alerte', 'orange', 'departement', 'l'ouest', 'ecole', 'inondation', 'bfmtv', 'frapper', 'etablissement', 'd'electricite'

(a) Descriptions croisées pour le corpus sur la tempête Alex

Id	Contenu
0	'civilian', 'death', 'humanitarian', 'provide', 'assistance', 'europarl_en', 'weapon', 'emmanuelmacron', 'hurt', 'osce_smm', 'russia'', 'count', 'leave', 'video', 'regular'
1	'world', 'want', 'putin', 'invade', 'first', 'news', 'country', 'stand', 'absolutely', 'join'
2	'potus', '#stopputin', 'nato', 'security', 'council', 'close', '#stoprussia', 'exclude', 'vonderleyen', 'eucopresident', 'aggression', 'provide', 'zelenskyyua', 'kyiv', 'stop'
3	'biden', 'make', 'take', 'even', 'like', 'call', 'right', 'back', 'well', 'year', 'president', 'tran', 'next', 'long', 'face', 'watch', 'situation', 'china'
4	'chuck1one', 'edmemphispimpin', 'trace_avp', 'selldateamdolan', 'yankeesknicks99', 'not47buddz', 'legionofknick', 'freekeith', 'stalling_e', 'mikefromqns7', 'like', 'many', 'learn', 'plan', 'force', 'care', 'killing', 'mean', 'today', 'live'
5	'russian', '#russia', 'force', 'military', 'ukrainian', 'attack', 'borisjohnson', 'meta', 'million', 'foreign', 'student', 'indian', 'limit', 'jm_scindia', 'share'
6	'people', 'defend', 'innocent', 'help', 'kill', 'please', 'stop', 'need', '#putin', 'weapon', 'ukrainian', '#russia', 'send', 'soldier', 'fight'
7	'think', 'know', 'trump', 'good', 'country', 'would', 'time', 'thing', 'invade', 'right', 'well', 'like', 'india', 'fool', 'nuclear', 'year', 'conflict'
8	'matchedNumber', 'give', 'state', 'military', 'year', 'even', 'canva', 'wetv', 'vidio', 'netflix', 'disney', '#tigray', 'spotify', 'jual', 'hotstar', 'vladimir', 'house', 'crime', 'find', 'direction', 'provocation', 'first'
9	'russia', 'invasion', 'support', 'fight', 'aggression', 'troop', 'churchill', 'conventional', 'protest', 'safe'

(b) Descriptions croisées pour le corpus sur la guerre Russo-Ukrainienne

FIGURE 5.11 – Contenu croisé des descriptions. Les mots en bleus apparaissent dans toutes les descriptions. Ceux en vert foncés apparaissent dans les descriptions issues de deux sélections de candidats différentes. Le vert clair indique les mots qui n'apparaissent que pour une seule sélection de candidat.

La figure 5.11 permet de visualiser la variété des tags utilisés pour une même description, à travers les différentes sélections. Cela permet d'une part de vérifier que notre approche permet bien de mettre en évidence des sous-évènements dans les données. Pour la tempête Alex, on peut voir que les interventions des pompiers ont été particulièrement remarquables dans des villages des Alpes-Maritimes, que les vents ont été très violents

dans le Morbihan ou encore que les crues ont fait des victimes. Du côté des données sur la guerre en Ukraine, on distingue des appels lancés à différents dirigeants ou à l'attention internationale.

5.3.1.5 Couverture des données

Nous considérons que pour être représentatives des données qu'elles concernent, les descriptions doivent en couvrir une partie significative. Nous présentons dans cette section la couverture obtenue selon la méthode de sélection et le jeu de données. Nous présentons la couverture "réelle", c'est à dire le nombre de tweets contenant un mot d'une description, et la couverture de la contrainte du modèle de l'équation 5.8.

Nous pouvons observer les différentes couvertures obtenues pour le jeu de données sur la tempête Alex dans la figure 5.12. La première chose que nous pouvons relever est que la couverture la plus élevée obtenue est de 41% pour la sélection par les scores DF-IDF. En considérant la variabilité dans les données en langue naturelle, c'est une proportion significative. Comme on peut s'y attendre, la sélection par fréquence obtient un résultat similaire étant également basée sur des informations statistiques. Autre observation attendue, les descriptions de candidats hybrides sont un peu moins couvrantes puisque l'utilisation de FPF dans cette sélection y limite l'impact de la fréquence.

Du reste, nous pouvons voir que certains clusters sont plus couverts que d'autres. Quelle que soit la sélection choisie, le cluster numéro 7 dans les données sur la tempête Alex est particulièrement bien couvert par rapport aux autres. Notons que la description 7 obtient les meilleurs scores d'évaluation pour ce corpus. Notons également que le cluster 3 est toujours le moins bien couvert, alors que la description 3 obtient de bonnes évaluations. Il semble donc que nos métriques d'IR et de SPD ne soient pas directement reliées à la couverture des clusters par leur description. De fait il semble qu'intégrer un critère de couverture s'appuyant sur une base différente de celle du DF-IDF (les statistiques) et de celle de la SPD (l'espace de plongement des mots) soit bénéfique pour la qualité des descriptions.

Deux autres clusters bien couverts sont les 6 et 9. Dans le tableau 5.11a, nous pouvons voir que ces descriptions sont à propos d'un sujet bien identifié. La description 6 semble indiquer que son cluster contient des messages à propos des personnes portées disparues à causes des crues dans les Alpes-Maritimes. Le 9 semble ainsi être construit avec des messages contenant des alertes météo dues à la tempête pour la Bretagne. Les descriptions obtiennent des scores d'IR et de SPD.

Il semblerait donc que les clusters les mieux couverts soient à propos d'un sujet que l'on peut séparer des autres. C'est le cas des 0, 6, 7 et 9. Constatons que le cluster 1 est plutôt bien couvert aussi. En regardant le contenu de la description associée, ce résultat s'explique du fait que les mots de sa description peuvent presque tous être trouvés dans différents contextes. `c'est`, `faire` et `meme` ne sont effectivement pas des mots qui soient spécifiques à un sujet en particulier, et sont donc ambigus sans plus de contexte.

Les descriptions de ces cinq clusters obtiennent toutes de bonnes évaluations d'IR et de SPD. Ainsi, il semble qu'un cluster ne sera pas bien couvert par une description qui obtiendra une mauvaise évaluation avec nos métriques. Cela étant, de bons scores avec ces dernières ne garantissent pas une bonne couverture.

La couverture que nous imposons avec notre contrainte est bien corrélée à la couverture réelle, ce que nous pouvons voir dans les tableaux 5.12 et 5.13. Leur valeur absolue ne

correspondent naturellement pas, mais les échelles des deux couvertures entre les clusters sont similaires. Il semble que si la valeur de couverture contrainte est élevée, alors celle de la couverture réelle le sera aussi. Il apparaît par ailleurs que pour la description hybride, les échelles de couverture réelle et par contrainte par cluster sont bien plus similaires.

Enfin, la taille des clusters ne semble pas être directement liée à leur couverture par leur description. Si les mieux couverts ne sont pas les plus grands, les plus petits ne sont pas nécessairement mieux couverts. Par exemple, le cluster 2 est le plus petit avec 631 tweets alors que le moins bien couvert est le 3 avec 2743 tweets.

La figure 5.13 sur la couverture des descriptions des données sur la guerre en Ukraine offre des observations similaires. Les clusters 2, 4, 6 et 7 sont les mieux couverts et obtiennent de bons scores d'IR et de SPD quelle que soit la sélection de candidats descripteurs employée. Le cluster 4 reste un cas particulier en raison de sa taille très réduite et du fait qu'il nous soit impossible de trouver un sujet commun à ses messages.

Il est plus difficiles de comprendre un sens à partir des descriptions 0, 1, 3, 5 et 8 dans le tableau 5.11b, leurs évaluations en IR et SPD étant plus mauvaises. Les clusters associés sont moins bien couverts. Cette corrélation est la même que pour les données sur la tempête.

La principale différence observée est la dé-corrélation des échelles de couverture contrainte et réelle pour les sélection par fréquence et par le DF-IDF. Elles concordent toutefois pour la sélection hybride, ce que l'on retrouve dans l'autre jeu de données.

Il semblerait donc que la différence de langue et de sujet aussi aient un effet notable sur la construction de description à partir de candidats identifiés sur la base de leurs statistiques.

FIGURE 5.12 – Couverture des données du corpus sur la tempête Alex, par cluster et pour le corpus, par sélection de candidat.

FIGURE 5.13 – Couverture des données du corpus sur la guerre Russo-Ukrainienne, par cluster et pour le corpus, par sélection de candidat.

5.3.2 Analyse générale de nos résultats

Des tendances communes émergent des trois méthodes de sélection des candidats que nous avons présentées. Une faible diversité parmi les candidats sélectionnés semble globalement détériorer les résultats, et certains clusters semblent généralement plus faciles à décrire que d'autres, quelle que soit la sélection. Par exemple, le cluster 7 dans l'ensemble de données sur la tempête Alex et le cluster 2 dans l'ensemble de données sur la guerre en Russie-Ukraine sont toujours bien identifiés, tant en IR qu'en SPD. En revanche, le cluster 5 pour Alex et le 9 pour l'Ukraine posent toujours des problèmes au modèle de description. Le même problème se pose avec la couverture, avec des clusters plus difficiles à couvrir, comme nous pouvons le voir dans les Figures 5.12 et 5.13.

Plusieurs questions se posent sur la difficulté pour un être humain à comprendre les descriptions des clusters. Par exemple, les scores d'IR et de SPD de la description du cluster 7 sont toujours bons, quelle que soit la sélection des candidats. Elle contient toujours les mots suivants : `vent` et `kilomètre`. `rafale` apparaît avec la fréquence et le DF-IDF, `côte` avec la fréquence et l'hybride. Ce cluster semble se concentrer sur la première partie de la tempête, impliquant des vents dans le nord-ouest de la France, mais pas d'inondations, par exemple, comme cela a été le cas dans le sud-est, dont les dommages semblent être capturés dans les clusters 0 et 6. L'hypothèse distributionnelle semble se vérifier dans la construction de l'espace de plongement et permet de comprendre facilement ce genre de cluster. Les vecteurs des tweets contenant `disparu` et `personne` semblent suffisamment proches les uns des autres pour être toujours groupés ensemble. En examinant les descriptions qui ont toujours les mêmes scores quelle que soit la sélection des candidats, nous pouvons avoir une idée de la tempête Alex.

En revanche, pour le cluster 4, les choses sont moins claires. Ses meilleures notes en SPD ne sont jamais sur la diagonale et ses lignes correspondantes dans les matrices d'IR sont toujours assez colorées. Cette description n'est pas bien isolée. En examinant les descripteurs, aucun événement majeur ne peut être interprété. Les mots communs aux trois sélections sont `plus` et `jour`, ce qui est dénué de sens car nous ne pouvons pas les associer à un autre mot commun. Nous avons également `fait`, `aussi` et `merci` deux fois sur trois, mais cela n'apporte pas plus d'informations.

En fonction de l'ambiguïté d'un mot, il peut être facile ou difficile de comprendre ce qu'il signifie en dehors de son contexte. C'est le cas de ce que nous appelons les mots outils. Par leur nature et parce que ces mots apparaissent dans de nombreux contextes, les vecteurs contenant ces mots sont naturellement plus éloignés des zones de l'espace sémantique correspondant à un sujet particulier. Les clusters peuvent alors être plus difficiles à expliquer en raison de leur composition.

Le jeu de données sur la guerre en Ukraine montre un autre phénomène que celui de la tempête Alex. Les trois méthodes de sélection ont identifié moins de 100 candidats, allant même jusqu'à moins de 80 pour la méthode par fréquence. Cela reflète une très grande redondance dans les données et un vocabulaire relativement restreint utilisé sur la plateforme. Cela pourrait être dû à la nature de l'événement écouté lui-même. Tous les types d'événements n'exigent pas la même variété de mots pour être abordés, que leurs sous-événements soient vraiment différents les uns des autres ou non. Notre approche pourrait donc avoir besoin d'être spécialisée pour détecter des parties plus fines d'événements "importants" moins variés, bien qu'elle puisse toujours en détecter certaines. Une autre explication pourrait venir de la dynamique de l'évènement écouté. Une tempête est un

évènement ponctuel, dont on relate tous les effets en un temps relativement court. La guerre en Ukraine est toujours en cours au moment de la rédaction du présent manuscrit. La façon dont les utilisateurs de Twitter parle de ces deux choses est donc nécessairement différente.

Il apparaît que le principal problème rencontré pour décrire les deux jeux de données vient de la difficulté à isoler des objets sémantiques, les sous-évènements que nous cherchons. Comme les plongements sont statiques, les différents contextes dans lesquels apparaissent les mots sont encodés dans un même vecteur. En fonction du vocabulaire employé pour parler d'un sous-évènement, il peut ainsi être particulièrement difficile à isoler dans l'espace des documents.

Cette ambiguïté cause également des problèmes d'interprétation des descriptions obtenues. Presque tous les mots peuvent être interprétés différemment selon leur contexte. Sans aucune information sur ce dernier, les descriptions peuvent ainsi être difficile à comprendre. Le meilleur exemple dans notre cas est celui du token que nous utilisons pour remplacer les nombres, `matchedNumber`. Lorsqu'il est retenu dans une description, s'agit-il d'une vitesse, d'un quantité ou d'une distance ? Dans la ligne 7 du tableau 5.11a, on voit `matchedNumber`, `kilometre` et `heure`. Dans le cas de ce cluster, ce `matchedNumber` est une vitesse, ce qui correspond à l'interprétation la plus directe lorsqu'on voit ces trois tags ensembles. Toutefois, rien ne garantit dans la sélection de candidat que `kilometre` et `heure` soit bien lié à `matchedNumber`.

Une manière efficace d'améliorer notre approche serait de désambiguïser les mots des descriptions en récupérant aussi une partie de leur contexte si besoin. Si ce changement peut s'opérer dans l'espace de représentation des mots, le clustering aurait par ailleurs plus de facilités à identifier des zones sémantiques distinctes.

5.3.3 Comparaison avec d'autres approches

Nous comparons notre méthode à BERTopic, présenté en section 3.3.2.1, que nous avons utilisé pour identifier 10 sujets (Grootendorst, 2022). Nous avons également effectué des tests avec le modèle de LDA de (Hoffman et al., 2010), mais il n'est pas adapté à l'identification des sujets dans les clusters de tweets. Ce modèle ne parvient pas à identifier clairement les sujets dans nos clusters, la probabilité la plus élevée parmi les mots des sujets identifiés atteignant environ une très faible valeur de 0,016.

Nous avons donc laissé cette approche de côté pour nous concentrer sur d'autres concurrents.

Tout d'abord, BERTopic ne s'attaque pas à la même tâche que notre approche. Nous visons à décrire un ensemble de données entier à travers les descriptions de ses structures sous-jacentes. L'objectif de BERTopic est d'identifier uniquement des parties pertinentes d'un ensemble de données et de les qualifier à travers un sujet. Bien que les deux tâches puissent sembler similaires, elles ont en réalité un champ d'application différent en ce qui concerne les données.

Deuxièmement, les ressources utilisées sont différentes. BERTopic utilise un modèle linguistique pré-entraîné pour représenter la sémantique des données textuelles sur lesquelles il travaille. Il a donc une connaissance plus générale de la langue, car il a été formé sur un échantillon de discours plus large. De notre côté, nous formons notre modèle linguistique in situ, ce qui donne une représentation plus spécialisée ou locale des informations

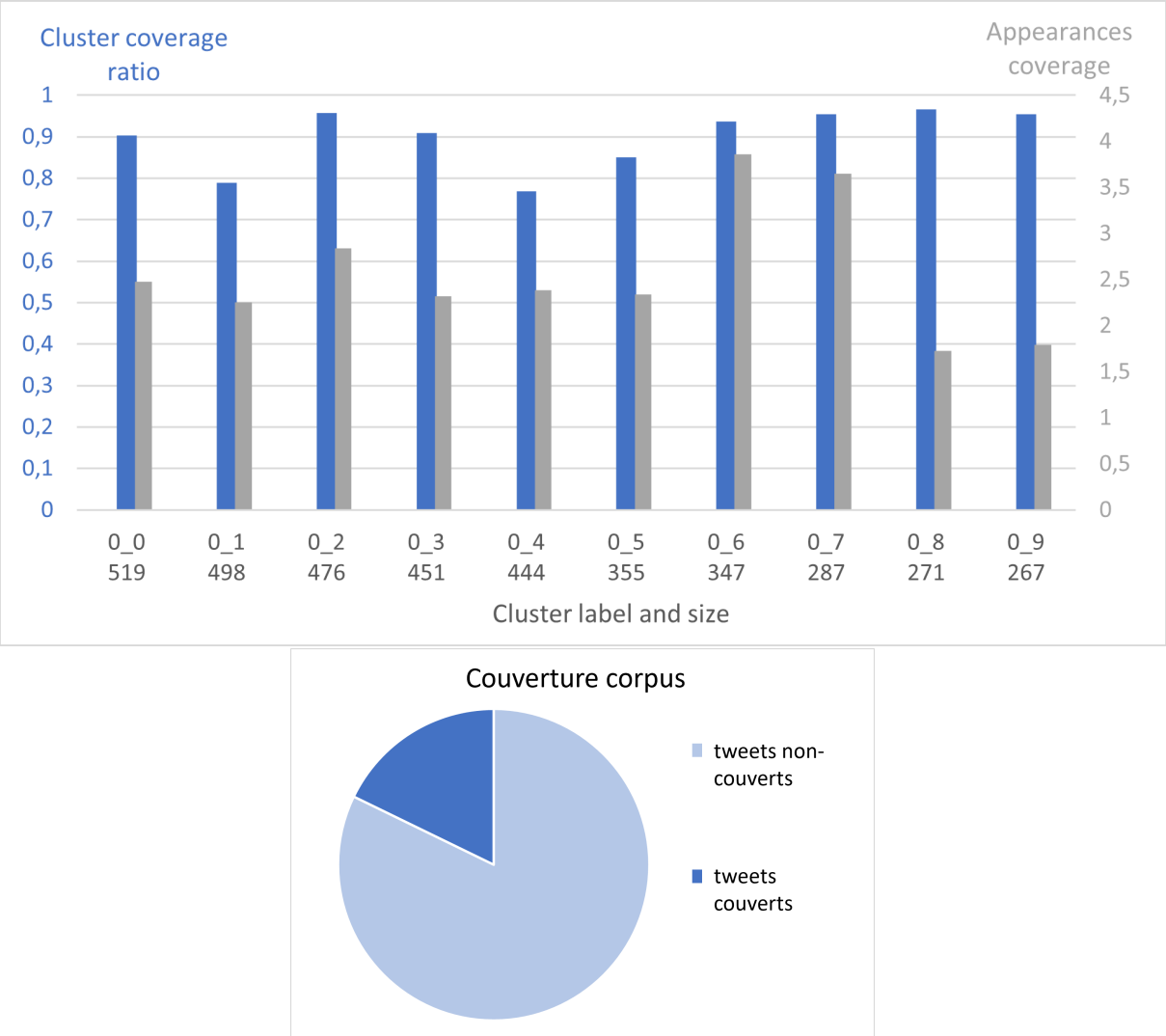


FIGURE 5.14 – Couverture du corpus par les mots de BERTOpic. 17.8% du corpus est couvert.

contenues dans l'ensemble de données. Cela conduit les deux approches à "comprendre" différemment les données avec lesquelles elles travaillent. Les deux approches sont également basées sur des lexiques légèrement différents en raison de leurs pré-traitements différents.

Il faut noter que même si les étiquettes de clusters de BERTopic et de notre approche sont les mêmes, elles ne font pas référence aux mêmes clusters. De plus, BERTopic ne calcule les plongements que pour les documents, pas pour leurs mots. Il n'est donc pas possible de calculer une évaluation de SPD pour les mots de ses sujets.

L'évaluation des sujets de BERTopic est excellente en terme d'IR, comme on peut le voir dans le tableau 5.15a. La couleur est presque absente des cellules du tableau qui ne sont pas sur sa diagonale.

Si on tient compte des documents considérés comme non-pertinents par BERTopic, l'évaluation de ses résultats est très différente. Dans la Figure 5.16a, la matrice est assez colorée et deux lignes ont leur meilleur score ailleurs que sur la diagonale. Il semble

Desc \ Cluster	0	1	2	3	4	5	6	7	8	9
0	9.1843	0.0777	1.5406	0.3846	0.9368	0.2174	1.2339	0.1336	0.0945	0.3025
1	0.4763	4.9853	0.9042	0.2868	0.6663	0.3141	2.9513	1.0387	0.7645	0.3545
2	2.7849	0.1579	10.5033	0.4469	0.1502	0.2438	1.2804	0.6651	0.1623	0.0368
3	0.3588	0.0502	0.2634	6.4364	0.4238	0.4837	0.4649	0.0723	0.2801	1.1021
4	0.3305	0.0978	0.1165	0.278	8.2533	1.8983	0.2355	0.0683	0.1918	0.3588
5	0.0846	0.0819	0.0991	0.7729	2.8875	5.5403	0.0654	0.1047	0.3816	0.0815
6	0.6928	1.1658	0.6295	0.2389	0.1852	0.0076	9.685	0.0749	0.2002	0.091
7	0.7934	0.6138	1.7826	0.1686	0.2716	0.177	0.4653	8.328	0.257	0.2793
8	0.1451	0.0075	0.0644	0.1673	0.1061	0.1139	0.2004	0.0872	14.4647	0.3568
9	0.2072	0.0245	0.0103	0.4985	0.2486	0.3424	0.2857	0.0152	0.1637	7.3699

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Desc	Words
0	vigilance, orange, département, rouge, météo, inondation, inondations, alerte, violent, pluie
1	ai, demain, cours, dehors, aller, est, rien, peur, cause, faire
2	kilomètre, par, heure, jeudi, rafale, météo, ouest, bretagne, vendredi, vents
3	image, dégât, vallée, montrer, roya, pont, maritimes, impressionnant, alpes, vésubie
4	personne, disparues, disparu, portées, deux, alpes, maritimes, dont, huit, maison
5	corps, pompier, retrouvé, vésubie, saint, martin, disparu, un, deux, sapeur
6	école, établissement, scolaire, fermés, cause, demain, lycée, morbihan, fermées, fermer
7	vent, semer, récolter, est, sème, récolte, vents, qui, souffler, qu
8	parc, loup, alpha, détruit, nature, loups, destruction, maritimes, alpes, vétérinaire
9	solidarité, collecte, dons, sinistré, aide, solidaire, nice06, alpesmaritimes, sinistrés, organise

(b) Description des clusters identifiés par BERTopic.

FIGURE 5.15 – Évaluation des descriptions des données tempête Alex par BERTopic.

donc que les mots des sujets puissent être utilisés pour décrire d'autres clusters que ceux auxquels ils sont assignés. Les clusters 1, 2 et 6 semblent bien identifiés, mais c'est moins clair pour les autres. Dans la Figure 5.14, on peut constater le principal inconvénient de cette approche. Seulement 19,6% des tweets du corpus sont couverts par les descriptions de BERTopic, ce qui représente plus de 20 points de moins que les descriptions Tops DF-IDF. BERTopic considère 16101 des 20016 tweets de notre corpus comme des valeurs aberrantes.

Desc \ Cluster	0	1	2	3	4	5	6	7	8	9
0	5.4716	0.9275	1.2613	0.7189	0.8366	0.4745	1.6753	0.7985	0.5204	0.7534
1	1.0988	2.0683	1.2117	0.8076	0.9743	0.7808	2.4731	1.0913	0.6598	0.8012
2	2.0938	1.383	4.7803	0.6796	0.5435	0.3339	1.5998	1.4319	0.5857	0.5156
3	0.4075	0.5534	0.3017	1.3617	1.2023	1.0266	0.4919	0.6847	0.8003	1.2611
4	0.5836	0.4947	0.4561	0.7764	1.9464	1.2904	0.5556	0.4509	0.9575	1.2749
5	0.2647	0.2654	0.3173	0.6486	0.9866	1.5746	0.2515	0.2681	0.8682	1.1242
6	1.0038	0.9894	0.6178	0.3585	0.3912	0.1531	4.4375	0.4379	0.3229	0.2794
7	0.8256	0.7231	1.1025	0.5271	0.3883	0.3377	0.543	1.5789	0.2427	0.3762
8	0.3187	0.3153	0.1012	0.3883	0.9379	1.1599	0.3396	0.5113	1.98	0.7631
9	0.1872	0.2687	0.107	0.4541	0.3664	0.8918	0.0616	0.2532	0.8801	0.847

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

FIGURE 5.16 – Évaluation des descriptions des données tempête Alex par BERTopic, en tenant compte des données considérées hors-scope par le modèle.

5.4 Conclusion

Ce chapitre présente l'architecture que nous avons mise en place pour éprouver nos hypothèses de travail. Son premier objectif était de montrer que s'appuyer sur un espace de représentation sémantique pour chercher des sous-évènements dans un jeu de données textuelles permet de les détecter. L'autre hypothèse à valider était le fait d'utiliser directement le contenu des tweets pour en décrire les ensembles, à travers une sélection de candidats descripteurs et un modèle déclaratif.

Les résultats obtenus montrent que ce type de représentation permet d'isoler des sous-parties cohérentes en terme de sujet au sein des jeux de données. Cette constatation indique que le modèle de changement de représentation Doc2Vec permet bien de calculer des plongements représentant, au moins partiellement, le sens des documents. La fait de pouvoir distinguer ces sous-ensembles les uns des autres à travers leur contenu confirme que ce modèle obtient des vecteurs robustes avec un entraînement court sur un jeu de données restreint.

Le modèle d'assignation que nous avons proposé permet d'obtenir des descriptions permettant de rendre compte de ce dont parle les données. Certains aspects ne sont mis en évidence que par certaines des méthodes d'identification de candidats descripteurs que nous avons présentées. D'autres apparaissent quelle que soit la méthode de sélection de candidat employée. Il apparaît que les jeux de données contiennent donc des éléments faciles à décrire et d'autres plus difficiles à isoler.

L'évaluation de la couverture obtenue par les descriptions indique qu'une portion significative des jeux de données est couverte. On peut donc les considérer représentatives de leurs corpus.

Les évaluations des méthodes sur des corpus différents, en langue et nature d'évènement, permettent des conclusions similaires. Si certaines observations sont propres à chacun d'entre eux, les tendances générales que l'on peut tirer des résultats concordent. Des évaluations supplémentaires sur d'autres jeux de données rendraient ces conclusions plus robustes. Il est à prévoir toutefois que l'évaluation étant partiellement manuelle, elle aura un coût en temps de travail important.

L'évolution la plus directe à apporter à ces travaux est l'amélioration de l'identification

des descripteurs. Le manque de spécifité des mots seuls et l'ambiguïté qu'ils peuvent véhiculer sont pénalisant sur différents aspects. D'une part, ils rendent naturellement les descriptions plus difficiles à interpréter. L'objectif étant de les présenter à un utilisateur humain, c'est un point critique à aborder. D'autre part, si un mot est ambigu, encoder ses différentes significations dans un vecteur répercutera cette ambiguïté dans l'espace des mots et par construction dans l'espace des documents. Limiter l'ambiguïté des mots du vocabulaire des corpus est donc aussi un élément important pour l'identification des sous-évènements.

Nous abordons ces points dans le chapitre suivant.

Chapitre 6

Vers une meilleure sélection des descripteurs de cluster

Sommaire

Nous l'avons vu dans le chapitre 5, il est possible d'utiliser un score pour déterminer à quel point un mot est approprié pour décrire un ensemble de messages courts. Une des limitations de ce type d'approche est de n'identifier que des tokens comme descripteurs. En perdant tout le contexte dont ils sont tirés, ils deviennent fréquemment ambigus. Nous

proposons d'aborder ce problème est exploitant directement le modèle de plongement pour ne pas avoir à mobiliser une autre ressource. Ce faisant, la connaissance utilisée pour rendre le contenu de l'ensemble de tweets compréhensible est tirée directement des données elles-mêmes. Doc2Vec étant un modèle léger et peu profond, le mettre à profit reste peu coûteux.

L'identification de descripteurs constitués d'un seul mot limite leur spécificité (Yan et al., 2013; Culmer and Uhlmann, 2021; Gracianne et al., 2022). Par exemple, le mot "*can*" seul peut avoir différentes significations et est spécifié par son contexte, comme dans "*tin can*" ou "*can do*". Il est donc important d'associer les mots qui sont liés entre eux afin de former des unités linguistiques cohérentes dans un corpus donné. Dans ce chapitre, nous présentons la méthode que nous proposons pour essorer le modèle de plongement que nous utilisons pour répondre à ce besoin.

Par ailleurs, rappelons que Doc2Vec apprend les représentations des mots et des documents au cours du même processus. Si on peut étudier les interactions entre les vecteurs de mot et ceux de contexte, on peut également tirer de l'information de celles entre les vecteurs de mot et ceux de document. Ce chapitre présente également comment nous proposons de le faire.

6.1 Unités multi-mots et modèle de plongement

Comme le modèle apprend à reconnaître les associations mots-contextes dans les données, nous pouvons l'exploiter pour identifier des unités formées de plusieurs mots faisant sens ensemble.

6.1.1 Objets recherchés

Avant d'expliquer comment nous proposons de détecter les groupes de mots sémantiquement liés, définissions ce que nous cherchons.

Les expressions polylexicales sont constituées de plusieurs mots qui forment une unité de sens unique, allant au-delà de la somme de leurs parties individuelles. Ces expressions sont souvent idiomatiques et leur signification ne peut pas toujours être déduite de manière littérale. L'étude des unités polylexicales offre un aperçu de la complexité du langage et de la créativité sémantique dans les données en langue naturelle. Leur reconnaissance est une tâche active en recherche. Des efforts ont été déployés pour détecter et reconnaître des expressions polylexicales, ou *multiword expressions* (MWE) (Maziarz et al., 2015), mais dans la plupart des cas, le système de reconnaissance est formé à l'aide de ressources telles qu'un corpus annoté ou un lexique (Kanclerz and Piasecki, 2022; Calzolari et al., 2002). Il est donc soit difficile à obtenir puisque nécessitant une intervention importante, soit coûteux à maintenir pour la même raison. Cela n'est pas compatible avec la nature hautement dynamique des données des réseaux sociaux. Selon la dernière tâche partagée sur les MWE, PARSEME 2020 (Taslimipoor et al., 2020), les MWE qui ont été déjà vues sous une forme lors de l'apprentissage sont bien reconnues, même avec une variation de forme. Il existe en revanche un véritable défi dans la reconnaissance de celles qui n'ont jamais été rencontrées lors de l'apprentissage. Des techniques basées sur l'attention qui reposent sur une architecture neuronale profonde effectuant une segmentation de texte fine (Badjatiya et al., 2018) peuvent être utilisées. Cependant, il semble s'agir d'un objectif trop détaillé

pour de telles techniques, tout en nécessitant le déploiement d'une architecture de modèle profond bien plus coûteuse en termes de calcul et de ressources que Doc2Vec.

Notons que des modèles de plongements dynamiques, comme par exemple des transformers comme BERT (Devlin et al., 2018), répondent naturellement au besoin de désambiguïsation. En effet, ils construisent un vecteur pour chaque occurrence de chaque mot dans les données. Toutefois comme nous l'avons dit, nous ne souhaitons pas avoir recours à des modèles ou des ressources trop coûteux à déployer pour pouvoir suivre le dynamisme de la langue employée sur les réseaux sociaux. De plus, les travaux de (Ferret, 2022) montrent que les plongements statiques présentent un intérêt non négligeable pour le traitement des mots peu fréquents dans leur corpus. Comme le vocabulaire des jeux de données issus des réseaux sociaux est particulièrement changeant, ce point est particulièrement intéressant pour nos travaux.

Exploiter en détail le modèle de plongement que nous utilisons, Doc2Vec, est une solution qui permet de ne mobiliser qu'un modèle léger et adapté à notre cas d'application. C'est pourquoi nous proposons d'utiliser le modèle de plongement pour aborder au moins partiellement le problème de la reconnaissance des unités de plus d'un mot. Nous appellerons ces dernières UMMs par la suite. Par ailleurs, pour identifier les entités ayant un nom composé dans les données, la Reconnaissance d'Entités Nommées (NER) pourrait être utilisée, mais même les approches actuelles rencontrent des difficultés pour accomplir cette tâche sur des données de médias sociaux en raison de leur bruit (Aguilar et al., 2017). Nous avons expérimenté les modèles NER de spaCy sur nos jeux de données de tweets, produisant des résultats inutilisables pour la comparaison.

La méthode que nous proposons n'est pas pensée pour détecter des expressions polylexicales de manière exhaustive. Cette tâche est complexe et le niveau de raffinement de l'information qu'elle permet d'obtenir est trop développé pour notre besoin. C'est pourquoi nous parlerons bien dans ce chapitre d'unités multi-mots. Ces UMMs désignent des groupes de mots qui ont du sens ensemble dans nos corpus d'études mais ne sont pas forcément des entités polylexicales ou des expressions multi-mots à proprement parler. Leur vocation se limite à désambiguïser les tokens concernés, en mettant à profit le modèle déjà présent dans notre architecture.

6.1.2 Fonctionnement de Doc2Vec

Dans cette étude, nous avons utilisé l'architecture PV-DM (Paragraph-Vector Distributed-Memory) du modèle Doc2Vec, comme illustré dans la Figure 6.1. Il s'agit essentiellement du même modèle que Word2Vec dans son architecture CBoW de (Mikolov et al., 2013a), modifié pour qu'il puisse apprendre les représentations des documents en même temps que celles des mots. Comme Word2Vec, le modèle déplace une fenêtre d'apprentissage sur les documents qu'il doit apprendre, afin de considérer les mots dans leur contexte et d'apprendre à **associer les mots et les contextes**. Les contextes sont constitués de mots issus des voisins droits et gauches des mots, tous de la même taille paramétrée l. Ainsi, la taille de la fenêtre d'apprentissage est $2l$. l est ajustable pour s'adapter à la nature des données traitées (Lau and Baldwin, 2016). Doc2Vec dispose d'un vecteur supplémentaire dans son entrée qui est utilisé pour représenter le document dont le contexte en cours de revue est issu. La taille de la fenêtre d'apprentissage pour Doc2Vec est donc de $2l + 1$. Tous les vecteurs appris ont la même taille paramétrée t.

Il a été démontré dans (Levy and Goldberg, 2014) que le modèle d'échantillonnage négatif (*Negative Sampling*) de (Mikolov et al., 2013b) factorise implicitement une variante de la matrice d'Information Mutuelle Ponctuelle (IMP) entre les mots et les contextes à partir d'un ensemble de données. Doc2Vec fait de même, en ayant un document qui peut être considéré comme une mémoire conservant la trace des opérations effectuées pour cette factorisation dans un document donné.

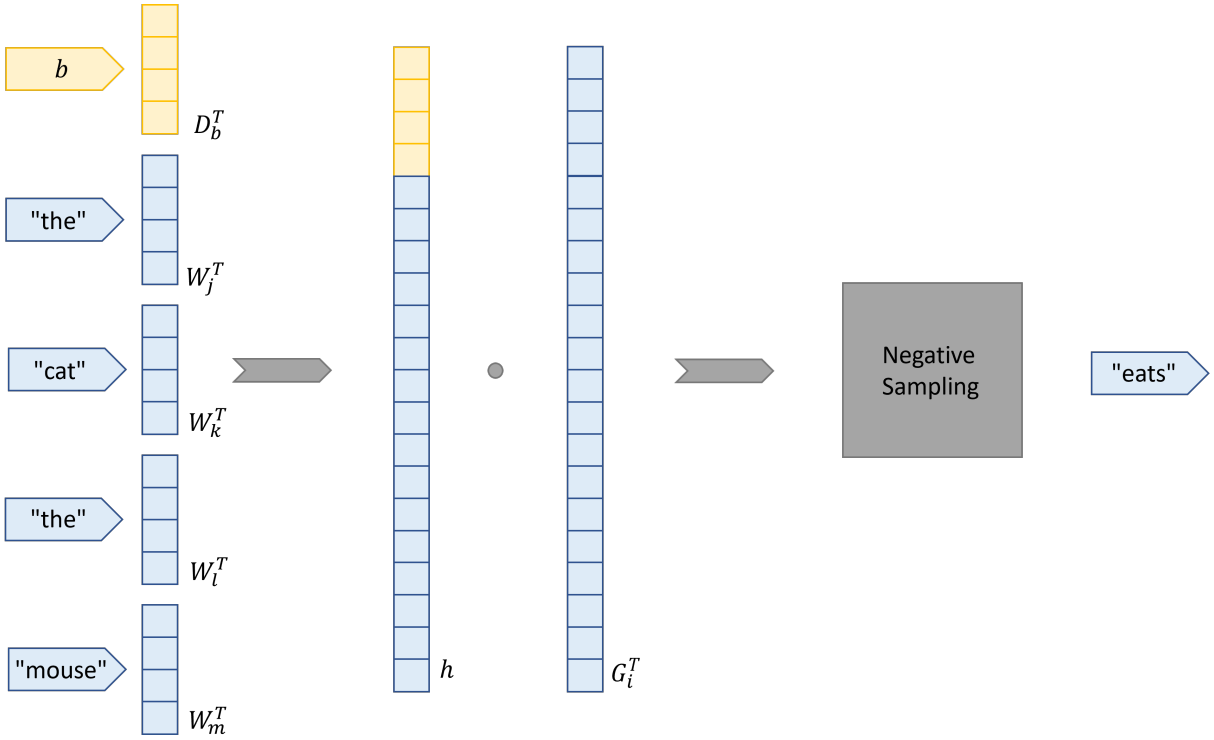


FIGURE 6.1 – Modèle Doc2Vec. Ici, le modèle tente de prédire le mot `eats` à partir de la séquence "the cat eats the mouse" apparaissant dans le document b. Il utilise les mots de contexte et les représentations de document ($\mathbf{W}_j^\intercal, \mathbf{W}_k^\intercal, \mathbf{W}_l^\intercal, \mathbf{W}_m^\intercal$ et $\mathbf{D}_b^\intercal$) assemblés dans le vecteur h et la représentation de contexte de "eats" ($\mathbf{G}_i^\intercal$) qu'il apprend.

Pour effectuer cette factorisation, pour chaque prédiction d'une paire mot-contexte **réelle** observée dans le corpus, l'échantillonnage négatif effectue des prédictions pour k_{NEG} paires mot-contexte **fictives** appelées échantillons négatifs. Le modèle s'entraîne pour maximiser les chances de prédire un mot dans un contexte où il est observé et minimise ses chances de prédire un mot dans un contexte où il n'apparaît pas.

Soit $\mathcal{D}$ l'ensemble de N documents, $\mathcal{V}$ le vocabulaire avec V entrées uniques. En tant que représentations internes, le modèle apprend trois matrices :

— $\mathbf{D} \in \mathbb{R}^{t \times N}$ pour les documents
— $\mathbf{W} \in \mathbb{R}^{t \times V}$ pour les mots
— $\mathbf{G} \in \mathbb{R}^{t(2l+1) \times V}$ pour les contextes

L'entraînement de Doc2Vec dans l'implantation [1] que nous utilisons suit l'algorithme 13. Chaque vecteur de mot $\mathbf{W}_i^\intercal$ est associé au vecteur de contexte $\mathbf{G}_i^\intercal$. Nous pouvons voir à la ligne 9 que la mise à jour des vecteurs de document et de mot est calculée en

1. `https://radimrehurek.com/gensim/models/doc2vec.html`

Algorithme 13: Itération d'entraînement de Doc2vec

input: l'index b du document passé en revue, l'index i du mot à prédire, les vecteurs de mot $\mathbf{W}^\intercal_j$, $\mathbf{W}^\intercal_k$, $\mathbf{W}^\intercal_l$ et $\mathbf{W}^\intercal_m$ voisinant $\mathcal{V}_i$ dans ce contexte tiré de $\mathcal{D}_b$, $\mathbf{D}$, $\mathbf{W}$ et $\mathbf{G}$

1 $J \leftarrow$ une liste contenant l'index du vecteur de contexte de $\mathbf{G}$ associé à $\mathcal{V}_i$ et k_{NEG} indices de contextes non associés à $\mathcal{V}_i$

2 $h \leftarrow concat(\mathbf{D}^\intercal_b, \mathbf{W}^\intercal_j, \mathbf{W}^\intercal_k, \mathbf{W}^\intercal_l, \mathbf{W}^\intercal_m)$

3 $e \leftarrow vector(T, 0)$ /* e contiendra la mise à jour à appliquer aux vecteur de h */

4 **Pour chaque** $j' \in J$ **faire**

5 $f \leftarrow \sigma(h \cdot \mathbf{G}^\intercal_{j'})$ /* $f \in [0;1]$ est $P(w_{j'}|h)$ avec σ la fonction sigmoïde */

6 **Si** $j' = i$ **alors** $l \leftarrow 1$;

7 **sinon** $l \leftarrow 0$;

/* $j' \neq i$ signifie que la paire $(h, G^\intercal_{j'})$ n'existe pas dans $\mathcal{D}$ */

8 $g \leftarrow (l - f) \times \delta$ /* δ = taux d'apprentissage */

9 $e \leftarrow e + g \times \mathbf{G}^\intercal_{j'}$

10 $\mathbf{G}^\intercal_{j'} \leftarrow \mathbf{G}^\intercal_{j'} + g \times h$

11 $\mathbf{D}^\intercal_b \leftarrow \mathbf{D}^\intercal_b + e$

12 $\mathbf{W}^\intercal_j \leftarrow \mathbf{W}^\intercal_j + e$, $\mathbf{W}^\intercal_k \leftarrow \mathbf{W}^\intercal_k + e$, $\mathbf{W}^\intercal_l \leftarrow \mathbf{W}^\intercal_l + e$, $\mathbf{W}^\intercal_m \leftarrow \mathbf{W}^\intercal_m + e$

fonction du résultat de prédiction à la ligne 5. À partir de vecteurs initiaux initialisés à 0, le modèle met itérativement à jour ses représentations. Comme dans de nombreux autres modèles d'apprentissage, l'amplitude de ces mises à jour diminue à mesure que les époques d'entraînement avancent grâce au taux d'apprentissage décroissant δ. Pour chaque paire réelle de mot-contexte dans le corpus, le modèle calcule la mise à jour des vecteurs e en fonction de cette paire et de k_{NEG} paires fictives pour une itération d'entraînement avec :

$$e = \sum_{j' \in J} \begin{cases} \sigma(h \cdot G^\intercal_{j'}) \text{ if } i = j' \\ -\sigma(h \cdot G^\intercal_{j'}) \text{ else} \end{cases} \tag{6.1}$$

L'équation 6.1 résume la boucle à la ligne 4. J contient l'indice i du vecteur de contexte associé au mot devant être prédit et les k_{NEG} autres indices de vecteurs de contexte sélectionnés au hasard pour constituer les paires mots-contextes fictives. h est l'agrégation des mots de contexte comme illustré dans la Figure 6.2. Les vecteurs de document et de mots de contexte sont mis à jour par des sommes avec e. Le signe du résultat de σ est inversé s'il a été calculé pour un exemple négatif, et conservé tel quel pour l'exemple positif. C'est ce qui permet de regrouper les vecteurs des mots apparaissant dans un contexte similaire tout en séparant ceux qui n'apparaissent pas ensemble. C'est ainsi que le modèle capture, au moins partiellement, l'hypothèse distributionnelle (Harris, 1954).

Les vecteurs de contexte sont mis à jour en fonction des contextes examinés, assemblés dans h, grâce à des produits scalaires et des sommes. Notons que le vecteur h peut être constitué par somme ou par moyenne des vecteurs d'entrées. Si on utilise la moyenne, on perd l'information de la position de l'information issue de l'entrée. On ne peut donc ainsi pas la corréler finement à une information issue du contexte associé dans $\mathbf{G}$. Nous

formulons donc l'hypothèse suivante : une fois le modèle entraîné, nous pouvons identifier quelle partie de h est fortement pondérée et détecter à quel vecteur de mot dans $\mathbf{W}$ cette pondération est liée. Si l'hypothèse se vérifie, nous proposons d'utiliser les parties du contexte fortement pondérées pour identifier deux choses. D'une part les mots fortement co-influents et ainsi les UMMs, et d'autre part d'isoler les mots pertinents pour décrire un ensemble de messages des mots qui ne présentent pas d'intérêt pour cette tâche.

Durée de l'entraînement La durée de l'entraînement d'un modèle de plongement comme Doc2Vec se compte en "époque". Une époque correspond au nombre d'itérations d'entraînements que le modèle doit effectuer pour avoir passé en revue au moins une fois chaque exemple dans ses données d'entraînement. Avec notre modèle, cela correspond au nombre d'occurrences de mots dans le corpus puisque le modèle passe en revue chaque mot de chaque document.

À chaque itération sur une paire mot-contexte, le modèle met à jour les coordonnés des vecteurs. Comme nous l'avons vu, son objectif est de rapprocher les mots qui apparaissent dans des contextes similaires. Plus le nombre d'époques est grand et plus le modèle les fera converger. Les mots apparaissant dans des contextes différents étant éloignés les uns des autres dans le même temps, plus long est l'entraînement et plus la séparation des zones sémantique devrait être claire. Par ailleurs, comme les mises à jour s'opèrent par sommation, plus d'époques peuvent également mener à des distances plus grandes entre les mots.

Comme nous souhaitons exploiter le modèle pour désambiguïser les mots en plus d'obtenir des représentations de documents, il convient d'entraîner le modèle plus longtemps. Nous passons de 10 à 50 époques d'entraînement. Cela a un effet notable sur le clustering obtenu, comme nous le verrons en section 6.3.

6.2 Identification des UMMs et pertinence d'un descripteur

Nous obtenons une représentation sémantique des tweets qui est agnostique depuis leur initialisation, en apprenant nos représentations de documents et de mots pour chaque ensemble de données. Nos unités de mots sont constituées de termes lemmatisés sans mots vides. Ainsi, notre vocabulaire est aussi ajusté que possible à notre corpus. Cette section présente comment nous avons testé nos hypothèse sur les poids appris par le modèle pour en tirer parti.

6.2.1 Détecter les UMMs

Le modèle apprend les associations mot-contexte dans les ensembles de données en factorisant implicitement la matrice d'IMP reliant les mots et les contextes. Comme ces derniers sont appris sur la base de vecteurs de mots concaténés, l'importance d'un mot w_j dans le contexte d'un autre w_i peut être mesurée grâce au produit scalaire entre $\mathbf{W}_j^\intercal$ et la partie de $\mathbf{G}_i^\intercal$ correspondant à la position de w_j dans le contexte de w_i, que nous notons $\mathbf{G}_i^\intercal|_j$, comme illustré dans la Figure 6.2. Nous proposons de mesurer cette importance avec le score d'activation suivant :

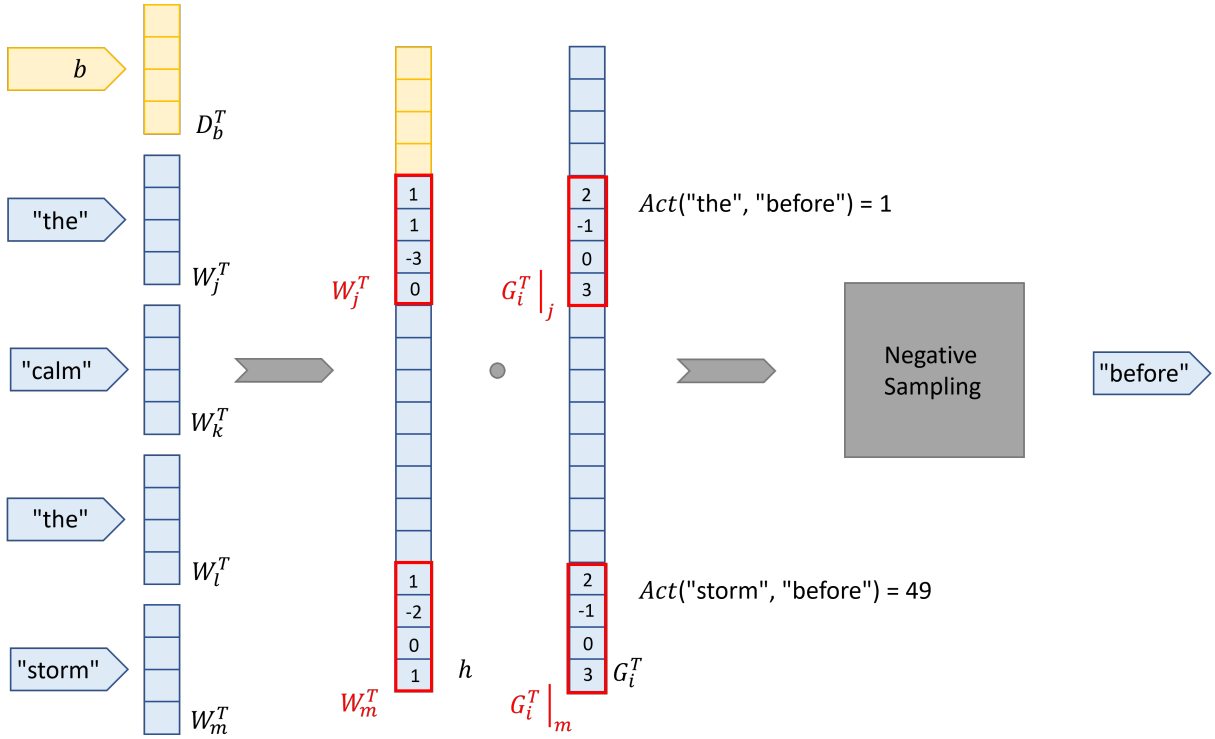


FIGURE 6.2 – Détection des mots fortement co-influents. Nous pouvons voir ici que "the" a une valeur *Act* beaucoup plus faible que "storm", par rapport à "before". Cela indique que, grâce à son entraînement, le modèle a appris que "storm" à la position m dans ce contexte est beaucoup plus utile pour prédire "before" que "the" à la position j.

$$Act(w_j, w_i) = (\mathbf{W}_j^\intercal \cdot \mathbf{G}_i^\intercal|_j)^2 \tag{6.2}$$

Cette mesure montre l'importance entre deux mots étant donné un contexte. Nous pouvons ensuite mesurer le score moyen *Act* entre deux mots w_i et w_j en considérant tous les contextes dans lesquels ils apparaissent ensemble. Cela permet de capturer à quel point ils sont importants pour se prédire mutuellement dans le corpus sur lequel le modèle est entraîné avec :

$$Act_{avg}(w_j, w_i) = \frac{\sum_{j \in J'} (\mathbf{W}_j^\intercal \cdot \mathbf{G}_i^\intercal|_j)^2}{|J'|} \tag{6.3}$$

avec J' l'ensemble des indices des contextes de prédiction de w_i dans lesquels w_j apparaît.

Si $Act_{avg}(w_j, w_i)$ est élevé, cela signifie que w_j a souvent été observé à la même position dans les contextes de w_i. Le modèle a donc appris à lui donner beaucoup de poids pour prédire w_i. Si cette relation est mesurée dans les deux sens, c'est-à-dire si à la fois $Act_{avg}(w_j, w_i)$ et $Act_{avg}(w_i, w_j)$ sont élevés, alors nous appelons w_i et w_j une **unité**. Si une unité a des valeurs de Act_{avg} significativement plus élevées entre ses parties qu'avec d'autres mots de ses contextes, nous l'appelons alors une UMM, car ses parties sont des mots **fortement co-influents**. Empiriquement, nous avons constaté que la plupart du temps il s'agit d'expressions multi-mots complètes ou partielles, de collocations ou d'entités nommées, telles que décrites par (Bender and Lascarides, 2020). Selon ces travaux, les expressions multi-mots (EMMs) sont des "collections de mots qui co-occurrent et sont

idiosyncratiques dans un ou plusieurs aspects de leur forme ou de leur signification. L'idiosyncrasie peut être simplement statistique, où les mots co-occurrent avec une fréquence beaucoup plus grande que le hasard". Ce sont des unités constituées de mots qui émergent de l'usage linguistique lorsque les mots sont souvent utilisés ensemble. Ils désignent des objets de discours ou des processus qui peuvent être aussi éphémères qu'une tendance sur les médias sociaux et qui peuvent constituer d'importants descripteurs de clusters de tweets pour un ensemble de données en particulier.

Pour détecter les UMMs, nous devons détecter les unités ayant des valeurs de Act_{avg} nettement plus élevées que les autres. Une valeur d'activation d'unité est

$$\frac{Act_{avg}(w_j, w_i) + Act_{avg}(w_i, w_j)}{2}$$

et toutes les unités identifiées sont classées en fonction de cette valeur.

Les membres d'une UMM qui apparaissent consécutivement dans le document sont fusionnés afin que les entraînements ultérieurs les considèrent comme un seul token. En répétant ce processus, il est possible d'identifier des UMMs plus longues. Comme le modèle est peu profond et léger, les entraînements successifs du modèle ne sont pas coûteux. Nous proposons de paramétrer le **seuil** d'activation qu'une unité doit atteindre pour être considérée comme une UMM, afin qu'il puisse être adapté. En considérant une paire de tokens, plus le seuil est bas, plus leur association doit être faible pour qu'ils deviennent une UMM.

Cette force dépend à la fois de la fréquence et de la spécificité de leurs co-occurrences. Les mots apparaissant toujours dans les mêmes contextes devraient avoir une forte association, et réciproquement pour ceux apparaissant uniquement dans des contextes différents. Un seuil élevé permet de fusionner uniquement les paires d'unités bien identifiées, avec le risque de passer à côté de celles ayant des associations légèrement moins fortes.

Les entraînements successifs pourraient être arrêtés lorsqu'il n'y a plus de tendance exponentielle parmi les valeurs de Act. Mais à ce stade, les unités pourraient couvrir des messages complets, laissant le modèle de plongement sans éléments de construction de documents. Pour nos deux ensembles de données, les unités multi-mots sont limitées à au plus quatre mots (non vides) de long. Dans ces langues, il est peu probable d'avoir des associations de mots plus longues dans des messages courts.

6.2.2 Étude par ablation de mot

Le second objectif est d'identifier des **descripteurs appropriés** pour un ensemble de messages au sein de leur texte. Nous proposons d'exploiter les espaces de représentation des documents et des mots pour ce faire.

Les vecteurs de documents sont appris de la même manière que les vecteurs de mots. Cependant, les mises à jour du vecteur d'un document sont basées uniquement sur les mots contenus dans ce document, tandis que les vecteurs de mots sont mis à jour en fonction de tous les contextes du corpus. L'effet d'une mise à jour est donc largement moins lissé dans les vecteurs de document et donc plus visible que pour les vecteurs de mots.

En nous basant sur le fonctionnement du modèle tel que présenté dans la section 6.1.2, nous émettons l'hypothèse suivante. Si un mot w apparaît toujours dans le **même voisinage**, la mise à jour basée sur l'erreur de la prédiction de w sera de **moindre magnitude**

que s'il est rare dans ce voisinage. Intuitivement, il est assez facile de comprendre que le modèle a moins de changements à apporter pour capturer une association mot-contexte très courante.

Si cette hypothèse est vraie, son effet sera observable dans l'espace de plongement grâce à une étude par ablation. Pour un document donné $d \in \mathcal{D}, d = [w_0, ..., w_i, ...w_n]$, soit $\vec{d}$ le vecteur inféré par le modèle pour représenter d et $\vec{d'}$ le vecteur inféré pour d lorsque le mot w_i est remplacé par le mot `NULL`. Ainsi, $dist(\vec{d}, \vec{d'})$ montre l'impact du mot w_i sur la représentation de son document. De faibles valeurs de ce décalage devraient indiquer des mots apparaissant dans des contextes similaires dans le corpus et, réciproquement, des valeurs élevées seraient associées à des mots apparaissant dans des contextes divers.

Toutefois, les valeurs que pourraient prendre ces décalages resteraient ininterprétables telles qu'elles. Elle sont en effet des mesures de distances dans un espace de haute dimensionnalité et n'ont pas de sens pour un humain. Il faut donc identifier des points de repères dans cet espace pour pouvoir tirer de l'information de ce décalage. Comme nous travaillons dans un ensemble de documents, nous proposons d'exploiter ce dernier.

Soit $\vec{c}$ le point représentant $\mathcal{C}$ dans l'espace de plongement, par exemple le vecteur moyen de tous ceux contenu dans le cluster. Soit $F = \{d \in \mathcal{C} : w \in d\}$ et b le nombre d'occurrences de w dans $\mathcal{C}$. Nous avons :

$$I(w, \mathcal{C}) = \frac{\sum_{d \in F}(dist(\vec{d'}, \vec{c}) - dist(\vec{d}, \vec{c}))^2}{b} \tag{6.4}$$

$I(w, \mathcal{C})$ est ainsi le décalage moyen induit par w par rapport à $\mathcal{C}$. Cette mesure corrèle ainsi la topographie de l'espace de représentation de l'ensemble de documents traités et la magnitude de l'impact des mots sur leur document. Nous proposons de l'utiliser comme base afin de discriminer les unités pertinentes pour décrire l'ensemble $\mathcal{C}$ de celles qui n'ont pas d'intérêt en tant que tags de description.

6.2.2.1 Mots sans intérêts descriptifs

Différents types de mots sont considérés comme des valeurs aberrantes compte tenu de l'objectif final de description. Nos jeux de données sont collectés à l'aide de requêtes par mots-clefs et sont donc déjà restreints autour d'un sujet. Cependant, en raison de la grande diversité de contenu et de sources sur les réseaux sociaux, les corpus ne contiendront pas seulement des informations sur le sujet cible.

Nous faisons le choix de ne pas chercher à détecter des sujets au sein des données pour les raccrocher à la cible. Si la requête a capté un contenu qui n'était pas prévu, nous considérons qu'il faut le décrire et laisser à l'utilisateur le choix de le considérer pertinent pour lui ou non. Si ce contenu n'apparaît pas en assez grand nombre dans l'espace, il sera par construction masqué dans le reste du jeu de données.

Nous considérons comme **sans intérêt descriptif** les mots qui n'ont aucune valeur en tant que descripteurs d'ensemble de message. Nous émettons l'hypothèse qu'ils ont des **valeurs extrêmes de** I. Un $I(w, \mathcal{C})$ très faible devrait indiquer un mot apparaissant dans des contextes très similaires parmi $\mathcal{C}$, ce qui en fait un mot non représentatif de cet ensemble. À l'opposé, un $I(w, \mathcal{C})$ très élevé serait associé à un mot éparpillé dans une grande diversité de contextes dans $\mathcal{C}$, trop pour qualifier l'un d'entre eux. Les valeurs minimales et maximales de I utilisées pour identifier les valeurs extrêmes de l'ensemble

sont notées respectivement α' et β'. Comme on s'attend à ce que les valeurs de I suivent une tendance exponentielle, ces deux paramètres doivent être positionnés pour exclure les valeurs extrêmement basses et extrêmement hautes.

6.2.2.2 Descripteurs pertinents

Doc2Vec construit des représentations sémantiques. Ainsi, le "centre" de la représentation de l'ensemble de messages considérés correspond à son "contexte moyen". Les documents qui en sont les plus proches devraient également être les plus représentatifs de la zone de similarité contextuelle que cet ensemble représente.

De fait, les mots **ayant une valeur I comprise entre α' et β' et qui apparaissent dans les γ documents les plus proches de** c sont proposés pour décrire $\mathcal{C}$. La valeur de γ indique indirectement à quelle distance un document est autorisé à être du "contexte moyen" de l'ensemble. Notons que chaque mot ici peut également être une UMM si leur identification a été effectuée.

6.3 Résultats expérimentaux

Nous avons conduit les expérimentations, sur nos deux jeux de données, pour répondre aux questions suivantes :

— Les UMMs identifiées sont-elles utiles pour la description des clusters de tweets ? (voir la sous-section 6.3.4)
— Notre méthode proposée d'identification de descripteurs fournit-elle de bons descripteurs ? (voir la sous-section 6.3.5)
— Quelles sont les performances de notre méthode comparée à d'autres similaires ?

6.3.1 Temps de calcul

Toutes les expérimentations sont menées sur la même machine et les mêmes données de base.

Tableau 6.1 – Temps de calcul moyens des approches. Les durées sont approximativement les mêmes pour les deux jeux de données.

Ablation	BERTopic	BTM	DF-IDF
$\approx$ 5 minutes	$\approx$ 2 minutes 30	$\approx$ 13 heures	$\approx$ 2 minutes

Nous pouvons constater que toutes les approches s'exécutent à des vitesses similaires, à l'exception de BTM, qui nécessite des calculs beaucoup plus lourds pour identifier les distributions de probabilité. Le temps d'exécution de BTM augmente de manière dramatique avec la quantité de données à traiter, par exemple en s'exécutant en environ trois minutes sur un cluster d'environ 2000 tweets, par rapport à environ 13 heures pour l'ensemble du corpus de 20000 tweets.

En ce qui concerne l'allocation du temps, les deux tiers des 2 minutes et 30 secondes nécessaires pour BERTopic sont consacrés à l'ajustement fin de son modèle de plongement SBERT (Reimers and Gurevych, 2019), suivi de ses étapes rapides d'identification et

de fusion de clusters. La plupart du temps utilisé par le score DF-IDF est consacré à des opérations d'entrée/sortie. Notre méthode d'identification des descripteurs basée sur l'ablation est légèrement plus longue car elle doit examiner l'ensemble des paires mot-contexte de l'ensemble de données. Comme les descripteurs obtenus ont confirmé notre hypothèse, nous avons l'intention de réduire ce temps d'exécution en intégrant directement le calcul du décalage du vecteur de document par mot dans l'inférence de document de Doc2Vec.

6.3.2 Topographie des nouvelles partitions

Avoir augmenté la durée de l'entraînement change fortement la structure de la partition. Les vecteurs de mots et de documents étant plus "spécialisés", leur disposition dans l'espace change. Pour montrer ce changement, nous présentons les résultats de couvertures obtenus pour les descriptions faites de tags identifiés via le DF-IDF.

Pour simplifier la comparaison, rappelons les topographies obtenues précédemment dans les figures 6.3 et 6.4.

6.3.2.1 Sans UMMs

Nous pouvons voir sur la Figure 6.5 que les clusters ne sont plus dimensionnés de la même manière que lorsqu'ils étaient calculés sur des plongements entraînés pendant 10 époques. Leur taille est équilibrée et se chiffre en centaines de tweets, sauf pour deux clusters, le 0 et le 5 ici. Ces derniers concentrent plus des trois quarts des données. Notons que la couverture aussi change de comportement. Tous les petits clusters sont entièrement ou presque couverts par leur description. Les deux autres quant eux sont nettement plus difficiles à couvrir. Pour le cluster 0, le paramètre atteint par la paramètre α de la contrainte de l'équation 5.8 est même inférieur à 0,2.

Il semble donc que les groupes de documents soient plus resserrés autour d'un vocabulaire donné, ce dernier étant beaucoup plus épars pour les deux clusters plus volumineux. Les clusters plus volumineux semblent regrouper les documents qui ne relèvent d'aucun des objets des petits clusters.

La tendance observée au sein des données sur la tempête Alex se retrouve dans celles sur l'Ukraine. Dans la Figure 6.6, nous pouvons en effet voir un comportement similaire. Plus de la moitié des tweets est répartie entre les clusters 1 et 8, qui sont relativement mal couverts. Les huits autres ont des tailles variant de 586 à 1815 messages, et sont entièrement couverts par leur description.

Notons que la couverture globale des données passe de 41 à 46% pour Alex et de 35 à 52% pour l'Ukraine. Il semble donc que d'utiliser des représentations plus spécialisées des données permettent de construire une partition qu'il est plus facile de couvrir avec les descriptions que nous construisons. Cela peut s'expliquer par le fait qu'avec un entraînement plus long, les distances entre les différentes zones sémantiques dans l'espace augmentent. L'algorithme de clustering peut ainsi plus facilement les isoler les unes des autres. De plus, ces zones sont plus spécifiques à un vocabulaire donné, par construction des représentations des mots et documents. Elle sont donc respectivement plus simples à couvrir avec les mêmes tags, exception faite des clusters regroupant les documents qui ne font pas partie de ces zones.

FIGURE 6.3 – Couverture du corpus sur la tempête Alex, lorsque le modèle de plongement s'entraîne pendant 10 époques

FIGURE 6.4 – Couverture du corpus sur la guerre en Ukraine, lorsque le modèle de plongement s'entraîne pendant 10 époques

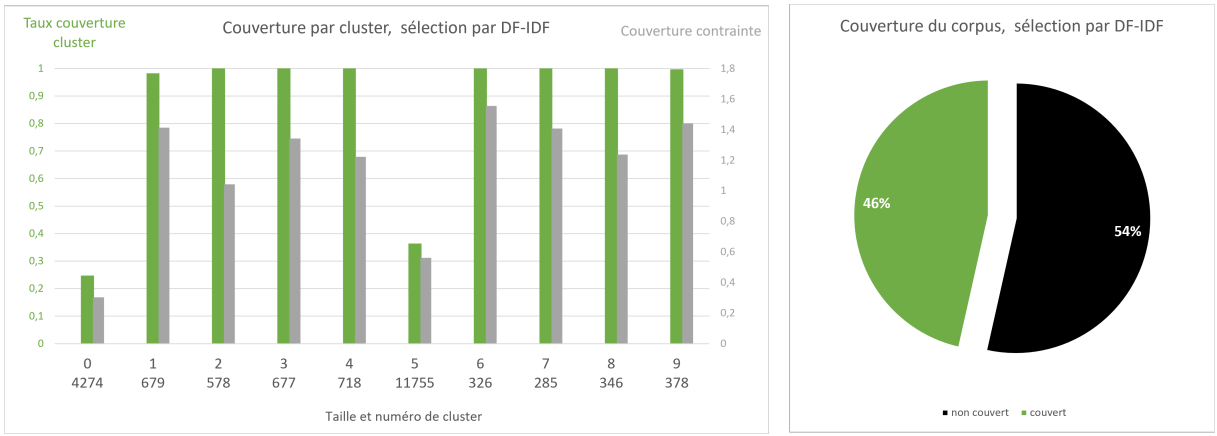


FIGURE 6.5 – Couverture du corpus sur la tempête Alex, lorsque le modèle de plongement s'entraîne pendant 50 époques, sans tenir compte des UMMs

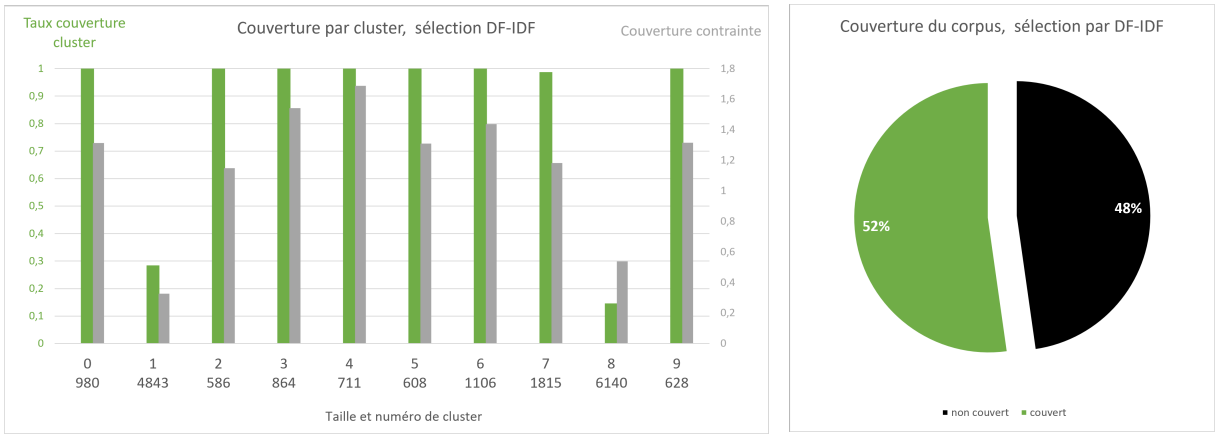


FIGURE 6.6 – Couverture du corpus sur la guerre Russo-Ukrainienne, lorsque le modèle de plongement s'entraîne pendant 50 époques, sans tenir compte des UMMs

6.3.2.2 Avec UMMs

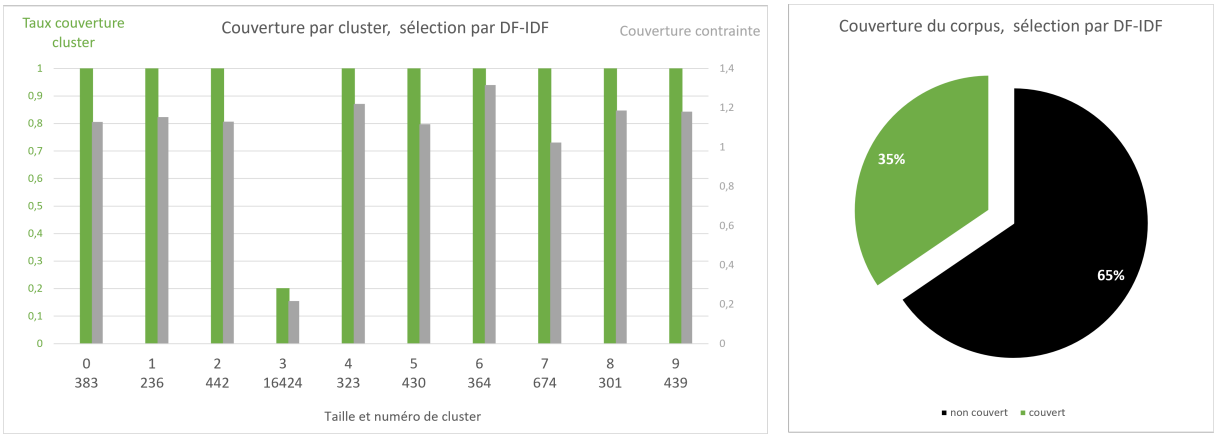


FIGURE 6.7 – Couverture du corpus sur la tempête Alex, lorsque le modèle de plongement s'entraîne pendant 50 époques, en tenant compte des UMMs

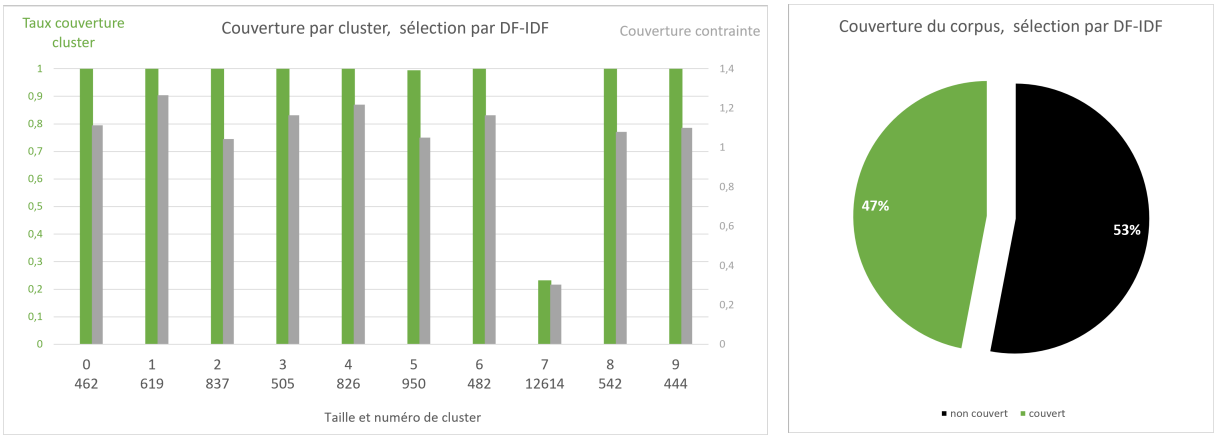


FIGURE 6.8 – Couverture du corpus sur la guerre Russo-Ukrainienne, lorsque le modèle de plongement s'entraîne pendant 50 époques, en tenant compte des UMMs

Les observations sur les partitions sur des plongements de 50 époques sans considérer les UMMs sont encore plus marquées lorsque les UMMs sont prises en compte. Nous pouvons le voir dans les Figures 6.7 et 6.8, à propos respectivement des données sur la tempête et sur la guerre. Dans les deux cas, un seul cluster couvre la majorité des données, plus des trois quarts avec le cluster 3 côté Alex et plus de la moitié avec le 7 côté Ukraine, et est assez mal couvert. Tous les autres clusters sont de taille similaire et sont entièrement couverts par leur description.

Deux interprétations sont alors possibles. L'apparition de nouvelles entrées de vocabulaire avec les UMMs pourrait avoir permis d'isoler plus finement encore certains aspects des données et explique ainsi la différenciation en 9 petits clusters. L'autre explication pourrait être qu'à l'inverse, les nouvelles entrées de vocabulaire ont "dilué" la sémantique capturée par les vecteurs, fondant ainsi ensembles les deux clusters de grande taille identifiés sans prendre les UMMs en compte.

6.3.3 Évaluation des résultats

Nous présentons ici l'évaluation en IR et SPD des descriptions obtenues avec notre méthode basée sur l'ablation de mots, tirant parti des UMMs.

Les tableaux 6.9a et 6.9b montrent une évaluation assez mauvaise avec nos critères des descriptions ainsi obtenues. La diagonale de la matrice d'IR est peu colorée et ne contient que 2 des 10 meilleures valeurs de ligne attendues. La SPD est un peu meilleure, avec 4 des 10 meilleures valeurs sur la diagonale.

La méthode de construction des descriptions basées sur la valeur de I semble donc montrer de mauvais résultats. Les mots des descriptions ainsi obtenus semblent peu distinctifs de leur cluster. 7 des 10 valeurs sur la colonne 0 de la matrice de SPD étant en orange, il semblerait également que les descripteurs pour un des clusters soient proches de tous les autres. Cet aspect peut s'expliquer par la présence d'un cluster de grande taille, relativement "centré" dans l'espace de plongement. Un pareil phénomène pourrait saper la consistance sémantique des descriptions des autres clusters.

Le tableau 6.9c permet toutefois de révéler l'intérêt de cette approche par rapport à celles que nous avons présentées précédemment. Les tags composés de plusieurs mots sont plus riches en information, et moins ambigus. Dans la description 0, nous pouvons en voir plusieurs : `entendre nature`, `alors arrêter`, `macron heure france`, `grand hotel`, `aller souffler fort`, `pouvoir faire` et `faire gaffe`. Il est difficile d'attribuer avec certitude un sens à `macron heure france` ou `entendre nature` sans le reste de la conversation. Dans le cas présent, ces tags sont apparus dans des discussions sur la réaction du président français face à la tempête et sur l'aggravation des effets de cette dernière du fait de la crise climatique mondiale. Ces descripteurs sont toutefois bien plus spécifiques que s'ils n'avaient été composés que d'un de leurs mots. Les autres tags composés sont plus faciles d'accès. `pouvoir faire` et `faire gaffe` sont des exemples de désambiguïsation du mot fonctionnel `faire`. Le premier apparaît souvent dans des questions rhétoriques comme "qu'est-ce qu'on peut y faire ?". L'autre est issu de conversation entre twittos, qui se disent de faire attention pendant la tempête. `aller souffler fort` est facile à interpréter en considérant que les données sont à propos d'une tempête et `grand hotel` est le nom d'une enseigne d'hôtellerie. Dans le cas présent, cela fait référence à un l'établissement de Belle-Île-en-Mer.

L'intérêt le plus visible se constate pour la description 5. Nous pouvons y lire `recherche poursuivre`, `chef sant-martin-vesubie`, `parc alpha detruit`, `porte disparu`, `loup nature`, `disparu alpes-maritimes` et `inquietude bilan definitif`. Ici les descripteurs composés permettent de facilement se faire une idée très claire de ce dont parlent les documents du cluster.

Ainsi, la sélection basée sur la valeur de I et la distance au centroïde permet de récupérer des descripteurs qui facilitent la compréhension des descriptions.

Toutefois, nous souhaiterions pouvoir garantir les mêmes contraintes de qualité que précédemment, tout en tirant parti de ces descripteurs plus riches de sens. Nous utilisons donc les descripteurs présentés dans la Figure 6.9 comme candidats et les donnons à notre modèle d'assignation. Nous obtenons les descriptions présentées, avec leur évaluation, dans les tableaux de la Figure 6.10.

Les évaluations en IR et SPD des descriptions ainsi construites montrent de bons résultats. Toutes les meilleures valeurs de lignes se situent sur la diagonale des tableaux 6.10a et 6.10b. Ainsi, il apparaît qu'une partie des descripteurs identifiés via leur valeur

de I et la distance au centroïde de leur document permet bien d'isoler les cluster et de former des descriptions cohérentes sémantiquement. Il faut pour cela les sélectionner en maximisant leur IR.

Toutefois, le tableau 6.10c montre que presque aucune des UMMs n'a été retenue de la sorte. En particulier, nous pouvons remarquer qu'aucune des UMMs ayant un sens clairement interprétable de la description 5 n'a été retenue. Il est ainsi difficile d'attribuer un sens aux descriptions obtenues de la sorte.

Les UMMs ne sont donc pas compatibles avec le score DF-IDF. Cela s'explique du fait qu'une unité composée sera naturellement plus rare en moyenne qu'une unité d'un seul mot. C'est particulièrement le cas quand elle comporte des mots fonctionnels. Comme le DF-IDF, utilisé pour calculer l'IR d'un mot dans un cluster, se base sur la fréquence, le découpage du décompte d'occurrence d'un mot entre ses différentes associations réduit mécaniquement le DF-IDF de celles-ci. Elles restent tout aussi rares à l'échelle du corpus, mais perdent en fréquence.

L'intérêt de la construction de description, via la valeur de I des mots et la distance de leur document au centroïde de leur cluster, ne pourra donc pas être mesuré avec nos métriques. Il faut donc inspecter les descripteurs identifiés pour les partitions pour comprendre l'intérêt et l'apport de cette méthode de description. Les résultats pour les données sur la guerre en Ukraine permettent les mêmes observations. Aussi nous les verrons avec les analyses qualitatives dans les sections suivantes.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	0.2127	0.1225	0.2222	0.2287	0.1754	0.3089	0.2441	0.13	0.2171	0.1626
1	0.2418	0.2547	0.1753	0.2432	0.3058	0.2824	0.3021	0.3276	0.3788	0.3545
2	0.549	0.6618	3.3981	0.2471	0.3952	0.3733	0.4634	0.5635	0.8281	0.5058
3	0.5472	0.5019	0.471	0.4222	0.4864	0.3658	3.775	0.6854	0.6616	0.7183
4	0.551	5.2755	0.5653	0.4091	0.6647	0.6358	0.7828	0.6697	0.6575	0.8794
5	0.1591	0.2958	0.3005	0.7453	0.4034	0.2108	0.344	0.2201	0.1844	0.2605
6	0.1941	0.3868	0.3125	0.3525	0.3199	0.2462	0.3034	0.1808	0.3331	0.3087
7	0.6339	0.4032	0.2672	0.2626	0.5222	1.8517	0.4868	0.436	0.5145	0.5056
8	0.7556	0.6675	0.6481	0.3391	0.6041	0.5469	0.6738	0.7575	4.6495	0.5998
9	0.1669	0.3671	0.1995	0.1806	4.135	0.4499	0.2301	0.2247	0.1913	0.3631

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	1.3898	1.1697	1.283	1.3611	1.355	1.3906	1.3592	1.3609	1.1711
1	0.8542	1.0	0.9232	1.0087	1.0535	1.0306	1.0213	1.0504	1.0456	0.8831
2	1.0243	1.3154	1.0	1.2717	1.3061	1.2854	1.2796	1.2753	1.2519	1.1068
3	0.8967	1.1471	1.0149	1.0	1.1395	1.1252	1.1524	1.1504	1.1359	0.9436
4	0.8617	1.0852	0.9443	1.0322	1.0	1.0493	1.0664	1.0729	1.0785	0.8883
5	1.0343	1.2798	1.1204	1.2288	1.265	1.0	1.1795	1.2808	1.2722	1.0908
6	0.9445	1.1286	0.9924	1.1199	1.1441	1.0495	1.0	1.1562	1.1244	0.9479
7	0.8675	1.0907	0.9295	1.0505	1.0816	1.071	1.0865	1.0	1.0601	0.9078
8	0.8633	1.0791	0.9068	1.0309	1.0805	1.0572	1.0501	1.0536	1.0	0.9075
9	1.1017	1.3517	1.1891	1.2701	1.32	1.3444	1.3131	1.3381	1.346	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'entendre nature', 'alors arreter', 'calmer', 'inadmissible', 'tenter', 'macron heure france', 'grand hotel', 'aller souffler fort', 'pseudo', 'pouvoir faire', 'vont', 'dechainer', 'faire gaffe', 'ptdrrr', 'reconstruire', 'resister', 'jsui', 'meteo'
1	'lumiere', 'meteo60', 'gere', 'bloque', 'bien soin', 'tenir', 'prend', 'premiere', 'arbre branche', 'priorite', 'obstacle', 'prevenir', 'va-t-elle', 'calme avant', 'merci', 'grace'
2	'expliquer', 'bretagne pays loire', 'senioush', 'bonjour', 'truc', 'esperer', 'ouverte', 'soule', 'ecouter', 'brasser', 'mdrrr', 'surprenant', 'courage breton', 'hein', 'aller', 'soeur', 'dendir35'
3	'sortir sous', 'didinechatninja ouestfrance', 'home', 'pont aerien', 'france', 'bete', 'entrainer', 'devenir', 'mois pluie heure', 'parvenir', 'mort disparu', 'deux mort', 'faire', 'jouer', 'eviter', 'envie', 'tempetueux', 'sortir', 'verite', 'peine', 'oreille'
4	'democrate', 'ridicule', 'quand venir', 'probable', 'indice', 'bombe meteorologique', 'episode', 'franceinter', 'prochain', 'certain', 'gouvernement', 'marche', 'virus', 'milieu', 'toucher', 'comme', 'pouvoir changer', 'tourner', 'prendre', 'ideologie', 'taux', 'francais', 'seum', 'grand'
5	'peut-on', 'apre midi', 'vivement', 'recherche poursuivre', 'chef saint-martin-vesubie', 'lemondefr', 'bois bout', 'alpes-maritime', 'fille', 'vive', 'parc alpha detruit', 'porte disparu', 'loup nature', 'disparu alpes-maritime', 'matchedHour', 'trois', 'inquietude bilan definitif', 'sinistre'
6	'laurent', '#cotedazurfrance', '#lorient', 'decoller', 'tres bien', 'prudent', 'puisse', '#saintlaurentduvar', 'envahir', '#cannes #cotedazurfrance', '#francemagique', 'restons', '#sea', 'errant', 'approche', 'apre', '#ploemeur', 'moral', '#cannes'
7	'projet', 'beau temp', 'article', 'connaitre', 'clone', 'drame', 'souhaite', 'pareil', 'meme', 'bfmtv', 'vigimeteofrance', 'ouragan', 'ptdrr', 'bosser', 'semer vent recolter'
8	'alpe maritime', 'forcement', 'horrible', 'cycliste', 'curieux', 'nostalgie', 'vivre bretagne', 'puisque', 'aussi', 'week-end', 'poussiere', 'absence', 'prevu', 'cnews', 'semer recolter', 'sinon', 'quand meme', 'bien', 'coucou', 'attend', 'mediavenir', 'degat chez'
9	'moto', 'rappeler', 'pleuvoir', 'christian', 'Alex', 'le_parisien', 'tres violent', 'rien voir', 'survecu', 'recap', 'cancun', 'mini', 'christ', 'matchedNumber', 'peser', 'clob755'

(c) Description des clusters

FIGURE 6.9 – Évaluation des descriptions sur les données sur la tempête Alex. Les descripteurs ont une valeur I non extrême et apparaissent dans les 15 documents les plus proches du centroïde de leur cluster.

Id description \ Cluster Id	0	1	2	3	4	5	6	7	8	9
0	0.6741	0.2776	0.2551	0.126	0.1534	0.2818	0.2245	0.306	0.2681	0.2584
1	0.3155	4.657	0.2088	0.1617	0.337	0.1804	0.2536	0.3886	0.3339	0.3646
2	0.34	0.6596	3.4719	0.2288	0.3329	0.4174	0.4861	0.3889	0.6786	0.5103
3	0.264	0.3508	0.3249	0.8046	0.4548	0.3513	0.2783	0.3013	0.3093	0.2756
4	0.2231	0.3351	0.2806	0.3036	4.6529	0.4282	0.3145	0.3619	0.3399	0.3775
5	0.4635	0.348	0.267	0.1891	0.2003	2.044	0.3165	0.2682	0.4792	0.3675
6	0.4399	0.536	0.3835	0.3829	0.3648	0.3782	3.9947	0.5563	0.4883	0.7169
7	0.3921	0.5362	0.342	0.2652	0.2266	0.2787	0.3833	0.6145	0.4296	0.3234
8	0.3387	0.323	0.3631	0.1948	0.3673	0.2515	0.4377	0.3819	4.5102	0.3344
9	0.3149	0.4459	0.4566	0.3836	0.3999	0.3932	0.5076	0.4046	0.5523	0.7914

(a) Matrice d'IR. Une cellule contient la valeur d'importance relative de la description de la ligne pour le cluster de la colonne. Plus la RI est élevée pour le cluster, plus la cellule est bleue.

Id description \ Description Id	0	1	2	3	4	5	6	7	8	9
0	1.0	1.2941	1.185	1.3045	1.2745	1.2743	1.2188	1.1547	1.294	1.1783
1	1.0602	1.0	1.0644	1.0958	1.1033	1.0945	1.0852	1.0609	1.1031	1.0717
2	1.0892	1.1942	1.0	1.212	1.1686	1.1789	1.14	1.1118	1.1445	1.1321
3	1.0876	1.1151	1.0993	1.0	1.1025	1.0986	1.1163	1.0815	1.1039	1.091
4	1.1123	1.1754	1.1096	1.1541	1.0	1.1335	1.1337	1.1211	1.1412	1.1017
5	1.06	1.1113	1.0669	1.0961	1.0803	1.0	1.0804	1.0619	1.1019	1.0653
6	1.0714	1.1644	1.0902	1.177	1.1418	1.1418	1.0	1.106	1.1174	1.1126
7	1.0687	1.1985	1.1194	1.2006	1.1889	1.1815	1.1645	1.0	1.1829	1.123
8	1.1489	1.1954	1.1055	1.1755	1.1609	1.1761	1.1286	1.1347	1.0	1.1367
9	1.057	1.1735	1.1048	1.1739	1.1323	1.1488	1.1353	1.0884	1.1485	1.0

(b) Matrice de SPD. Une cellule contient la somme pondérée des distances entre les mots de deux descriptions. Plus les mots sont proches les uns des autres, plus la cellule est bleue.

Id	Mots
0	'jsui', 'bete', 'verite', 'mediavenir', 'truc', 'ecouter', 'hein', 'puisse', 'lumiere', 'pareil'
1	'bombe meteorologique', 'episode', 'certain', 'marche', 'comme', 'seum', 'jouer', 'cnews', 'soeur', 'beau temp'
2	'prochain', 'virus', 'reconstruire', 'resister', 'sinon', 'expliquer', 'esperer', 'soule', 'aller', 'premiere'
3	'francais', 'vivement', 'lemondefr', 'alpes-maritime', 'trois', 'meteo', 'france', 'devenir', '#cotedazurfrance', 'semer vent recolter'
4	'matchedHour', 'eviter', 'alpe maritime', 'prevu', 'gere', 'moto', 'matchedNumber', 'souhaite', 'vigimeteofrance', 'ouragan'
5	'milieu', 'toucher', 'tenter', 'vont', 'dendir35', 'grace', 'le_parisien', 'tres violent', 'connaitre', 'meme'
6	'gouvernement', 'tourner', 'sinistre', 'calmer', 'faire', 'envie', 'peine', 'merci', 'rappeler', 'article'
7	'ridicule', 'prendre', 'horrible', 'aussi', 'mdrrr', 'tenir', 'calme avant', 'christ', 'bfmtv', 'ptdrr'
8	'oreille', 'vivre bretagne', 'week-end', 'quand meme', 'bien', 'coucou', 'bonjour', 'ouverte', 'moral', 'priorite'
9	'grand', 'home', 'sortir', 'puisque', 'prudent', 'apre', 'prend', 'pleuvoir', 'survecu', 'bosser'

(c) Description des clusters

FIGURE 6.10 – Évaluation des descriptions sur les données sur la tempête Alex. La liste initiale des candidats descripteurs est formée des unités ayant une valeur I non extrême et qui apparaissent dans les 15 documents les plus proches du centroïde de leur cluster. Cette liste est dé-doublonnée et passée au modèle d'assignation de tag au cluster.

6.3.4 Qualité des unités multi-mots

Il est délicat d'évaluer quantitativement notre détection d'UMM. En effet, une telle évaluation pourrait être effectuée en utilisant des jeux de données annotés pour la reconnaissance d'entités polylexicales. Il serait ainsi possible de voir combien d'entités sont reconnues et combien sont ratées. Toutefois, cette évaluation ne serait pas pertinente au regard de notre objectif. En effet, nous cherchons à exploiter à moindre coût les composants de notre approche pour identifier des descripteurs moins ambigus. Nous ne cherchons pas nécessairement à détecter toutes les entités polylexicales dans nos données. Ce serait une information plus fine que ce dont nous avons besoin.

Ainsi, pour être cohérente avec notre objectif, une évaluation quantitative devrait compter le nombre de fois où la détection d'UMM a amélioré la qualité d'une description. Hors, il n'existe pas de jeu de données de référence qui permettrait une telle évaluation. Pour estimer la validité de nos méthodes, nous avons donc recours à des évaluations qualitatives.

Les résultats présentés ici sont ceux après trois entraînements consécutifs du modèle de plongement. Les plus longues UMMs compteront donc quatre mots maximums. Les UMMs identifiées appartiennent majoritairement à deux catégories : des collocations, utiles pour la désambiguïsation, et des EMMs complètes ou partielles, de personne, d'organisation et/ou de lieu.

Si les UMMs rendent les descriptions plus accessibles pour un être humain, elles peuvent également être directement utilisées pour réduire la taille d'un ensemble de données. La fusion de leurs parties réduit le nombre de tokens à traiter par le modèle de plongement. Dans nos jeux de données, il a diminué d'environ 18,5% pour le corpus Alex (196890 à 160520) en identifiant 6234 UMMs et d'environ 13,8% pour l'ensemble de données sur l'Ukraine (235675 à 203194) avec 7908 UMMs. En raison de la légèreté et de l'architecture compacte du modèle, chaque itération d'entraînement et d'identification a pris moins d'une minute sur la même machine que pour les précédentes expérimentations.

6.3.4.1 Collocations

L'identification des UMMs montre son intérêt descriptif. Dans le cas d'événements venteux, par exemple, l'échelle de Beaufort peut être utilisée pour évaluer les effets du vent dans les terres, de 0 pour le calme à 12 pour les tempêtes et les ouragans. Le tableau 6.2 présente l'échelle au complet, telle que présentée sur Wikipédia [2]. Notons qu'en fonction de la source, la description des effets d'une tempête à un échelon change. Ces effets eux, restent les mêmes.

Le tableau 6.3 montre que certaines unités peuvent alors être directement utilisées, comme `arbre déraciner`. D'autres comme `chute branche` requièrent un passage par une ontologie pour certifier le correspondance entre leurs composants et les mots recherchés.

Ainsi, les UMs peuvent faciliter l'évaluation d'un événement en utilisant ce type d'échelle. Le jeu de données de la tempête Alex atteint le score maximal sur l'échelle de Beaufort, c'est-à-dire 12 pour les ouragans. Les UMMs présentées dans le tableaux sont issus de documents légèrement plus distants des centroïdes de clusters. Elles ressortent parmi les tags avec $\gamma = 17$.

2. `https://fr.wikipedia.org/wiki/Échelle_de_Beaufort`

TABLEAU 6.2 – Échelle de Beaufort et effets visibles à terre

niveau sur l'échelle de Beaufort	effet attendu
0	La fumée monte verticalement. Les feuilles des arbres ne témoignent d'aucun mouvement
1	La fumée indique la direction du vent. Les girouettes ne s'orientent pas
2	On sent le vent sur le visage. Les feuilles s'agitent. Les girouettes s'orientent
3	Les drapeaux flottent au vent. Les feuilles sont sans cesse en mouvement
4	Les poussières s'envolent. Les petites branches plient
5	Le tronc des arbustes et arbrisseaux en feuilles balance. La cime de tous les arbres est agitée. Des vaguelettes se forment sur les eaux intérieures
6	On entend siffler le vent. Les branches de large diamètre s'agitent. Les parapluies sont susceptibles de se retourner
7	Tous les arbres balancent. La marche contre le vent peut devenir difficile
8	Les branches sont susceptibles de casser. La marche contre le vent est très difficile, voire impossible
9	Le vent peut légèrement endommager les bâtiments : envols de tuiles, d'ardoises, chutes de cheminées
10	Dégâts importants aux bâtiments. Les toits sont susceptibles de s'envoler. Certains arbres sont déracinés
11	Ravages étendus et importants
12	Dégâts très importants de l'ordre de la catastrophe naturelle

TABLEAU 6.3 – UMMs identifiées corrélées à l'échelle de Beaufort

niveau sur l'échelle de Beaufort	effet attendu	UMMS identifiées correspondantes
8	"les branches sont susceptibles de casser"	`chute branche`
10	"certains arbres sont déracinés", "dégâts importants aux bâtiments "	`arbre arracher`, `arbre déraciner`, `dégât considérable`
11	"dégâts très importants"	`pont effondré`, `route coupée`

Dans nos données, nous avons par exemple `vallée ravagée`, qui n'est pas une association habituelle mais que le modèle a appris à fortement pondérer, de même que `venir aide sinistrée vallée`. Ces associations sont relatives à certains événements spécifiques rapportés au cours de la tempête Alex, où certaines vallées ont été particulièrement endommagées par la tempête. Dans la même catégorie, nous trouvons `disparu` associé à `rechercher` ou `supposer`, ainsi que `inondation` associée à `catastrophique` ou `meurtrier`.

Certaines des collocations détectées constituent des objets d'intérêt pour l'actualité à la date de publication des messages. Dans le jeu de données de la guerre en Ukraine, nous trouvons `sanction` associé à `tough`, `european` ou `nordstream`, des associations inhabituelles mais fortement pondérées dans ce corpus. Nous trouvons également `europe` associé à `shape`, `isolate` ou `solidarity`. Avec des unités plus longues, nous détectons `ukrainian foreign minister` et `declare state emergency`. Tous ces éléments sont relatifs aux évènement qui ont marqué le début de cette guerre.

L'identification d'unités plus longues permet de lever les ambiguïtés concernant des mots qui ne peuvent pas être interprétés seuls. Par exemple, dans nos données sur la tempête Alex, `sans` est utilisé de nombreuses manières différentes. Parmi les mots avec lesquels il est co-influent que nous détectons, nous pouvons voir `électricité`, par exemple, ou `oublier`. Dans le jeu de données sur l'Ukraine, nous avons `take care` et `take seriously`.

Toutes ces associations modifient le sens de leurs mots ou permettent de mieux comprendre leur contexte. Non seulement les mots `sans` ou `take`, pris seuls, ne fourniraient aucune information à un être humain s'ils étaient choisis pour décrire une zone de l'espace de plongement, mais les autres mots auxquels ils sont liés ne représenteraient pas non plus le contexte dont ils sont extraits. En effet, `take seriously` ne s'emploie pas de la même manière que `seriously` seul.

Nous constatons ainsi que l'identification des UMMs permet de créer des entrées de vocabulaire pour les associations de mots leur donnant un sens différent.

6.3.4.2 Entités nommées multi-mot

Si toutes les entités nommées désignées par un seul mot ne peuvent pas être identifiées avec notre méthode, elle s'avère efficace pour détecter les entités, de lieux et de personnes, désignées par un nom composé. Dans nos données, nous avons des exemples de personnes avec `emmanuel macron` et `donald trump`, toutes deux des unités composées de deux mots fortement liés l'un à l'autre. Nous avons `departement breton` et `ukrainian border` pour les lieux. Comme nous avons remplacé tous les nombres apparaissant dans le corpus par le jeton `matchedNumber`, le modèle est également capable de détecter des dates comme dans `matchedNumber september`.

Notre méthode détecte également des unités plus longues qui sont couramment utilisées dans les données. Par exemple, nous pouvons trouver `arriver bretagne jeudi soir`, ce qui correspond à la prévision de l'arrivée d'Alex annoncée pour un jeudi soir en Bretagne, ou `support no-fly zone` pour les mesures préventives correspondantes proposées au début de la guerre.

6.3.4.3 Amélioration du clustering

L'utilisation d'unités de texte plus longues améliore par ailleurs la qualité du clustering, selon des métriques standards telles que les indices de Davies-Bouldin et de silhouette. Le premier est calculé pour l'ensemble de la partition et ses valeurs vont de 1 (meilleure partition) à $+\infty$ (pire). Il diminue en moyenne de 15% (passant d'environ 2,07 à environ 1,77) entre les regroupements basés sur le plongement de tokens et ceux qui exploitent des unités de texte plus longues.

Les valeurs de l'indice de silhouette sont calculées pour chaque point (document dans notre cas). Elles vont de -1 (pire valeur pour un point) à 1 (meilleure valeur) et doivent être moyennées pour donner des informations sur l'échelle de groupement. Sur plusieurs exécutions de clustering, l'indice de silhouette s'améliore de presque 50% (passant d'environ 0,207 à environ 0,297) lors de l'utilisation de l'incorporation d'unités de texte plus longues.

6.3.5 Qualité des descripteurs

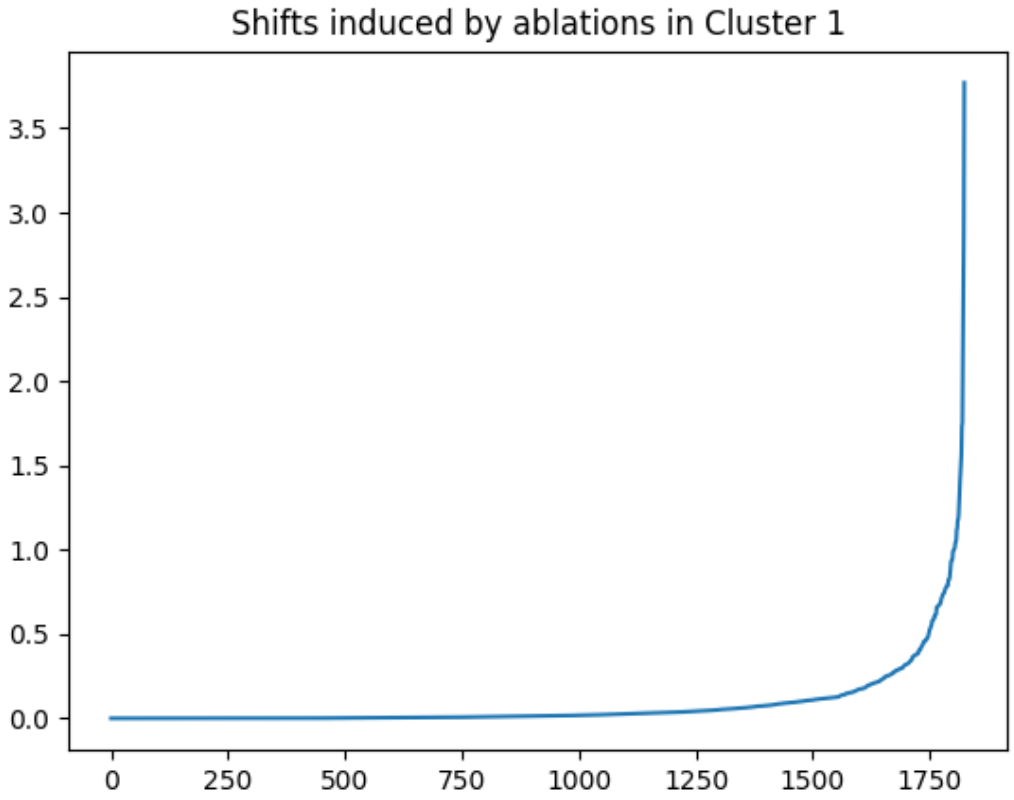


FIGURE 6.11 – $I(w, \mathcal{C}_1)$ pour tous les mots w d'un cluster tiré aléatoirement (le 1) dans le corpus Alex. L'axe vertical représente les valeurs de I, tandis que l'axe horizontal représente l'index des mots du vocabulaire du cluster, triés par ordre croissant de leur valeur de I.

En examinant les valeurs de I calculées pour chaque mot de chaque cluster dans les deux corpus, nous observons ce qui suit :

— Toutes les valeurs de I suivent une **tendance exponentielle** dans leur cluster, comme illustré dans la Figure 6.11, ce qui correspond au comportement attendu (voir la section 6.2.2) avec des mots apparaissant dans des contextes de diversité limitée et des mots apparaissant dans des contextes variés.

— Les mots porteurs de valeur descriptive se trouvent **entre ces deux extrêmes**.

Une unité considérée comme sans intérêt descriptif pour un cluster peut en avoir pour un autre cluster de la partition. Par exemple, dans le corpus Alex, le mot `delta`, le nom d'un ouragan qui a eu lieu en même temps que la tempête Alex, est considéré comme un élément atypique pour le cluster 4 tout en étant intéressant pour le cluster 5. Dans l'ensemble de données sur l'Ukraine, `jinping` est considéré sans intérêt pour le cluster 2 mais important pour le cluster 1. Il existe également des unités considérées comme atypiques pour tous les clusters, comme `astrology` dans l'ensemble de données sur l'Ukraine ou `capucine` dans l'ensemble de données sur la tempête Alex.

Nous corrélons les informations de I directement aux distances mesurées entre les documents et leur centroïde de cluster (voir la section 6.2.2) pour introduire davantage d'informations topographiques dans l'identification des descripteurs. Les deux sont des mesures de distance dans un espace de grande dimensionnalité, mais ils l'exploitent différemment, atténuant la perte d'information. Nous classons les documents du cluster en fonction de leur distance par rapport à leur centroïde de cluster et sélectionnons les $\gamma = 15$ plus proches. Les mots apparaissant dans ces documents avec une valeur de I entre α' et β' sont les descripteurs du cluster. Ici, α' est fixé au deuxième décile pour toutes les valeurs de I et β' au huitième. Ces seuils ont été identifiés empiriquement comme les plus adaptés pour les deux ensembles de données.

Décrire seulement avec les hashtags Pour vérifier si les hashtags issus des tweets puissent convenir pour décrire leur cluster, les 20 hashtags les plus fréquents sont extraits pour chaque cluster. Il apparaît que cette liste ne peut pas être considérée comme une description de partition. Dans l'ensemble de données sur la tempête Alex, cette liste est composée de 92 hashtags différents, dont 65 apparaissent deux fois ou moins dans leur cluster, respectivement réduits à 81 et 44 dans l'ensemble de données sur l'Ukraine. Cela vient du fait que les hashtags sont utilisés pour lier leur message à un objet, qui peut être un événement en cours, une tendance Twitter, une personne ou un lieu. Ils sont utiles pour étudier la dynamique de la plateforme, mais limités dans le cadre notre tâche.

De plus, parmi les hashtags les plus utilisés dans le corpus Alex, on trouve `#bretagne` et `#alpesmaritimes`, qui sont deux régions touchées par la tempête, et `#météo`. Dans le corpus sur l'Ukraine, ce sont `#russia`, `#putin` et `#nato`. Les mots de ces hashtags sont également présents sans le # dans les données. Même s'il existe des hashtags avec une utilisation spécifique à Twitter et qui sont constitués de mots inventés sans équivalents dans la langue en dehors des médias sociaux, ils sont beaucoup moins représentés. Nos observations nous encouragent à considérer les hashtags comme toute autre unité textuelle dans les données.

6.3.6 Comparaison des résultats

La plupart des méthodologies existantes permettant de décrire des ensembles de documents avec des unités de plusieurs mots ne sont pas accessibles, fautes de ressource pour les mettre en oeuvre. Parmi celle que nous pouvons envisager, il y a notamment l'extraction de phrase clef à partir de modèle thématique. Toutefois, nous n'avons pas trouvé de code accessible (récent ou non) pour des approches de ce type. Ainsi, nous employons à nouveau le modèle BTM de (Yan et al., 2013) et pour évaluer le gain de notre identification d'UMM, nous émulons la méthodologie d'extraction de mots-clefs proposée dans (Zhao et al., 2011a). Dans ces travaux, les phrase-clefs sont des combinaisons de mots clefs apparaissant fréquemment ensemble.

6.3.6.1 Comparaison à l'extraction de phrases clefs avec un modèle thématique

Nous avons exécuté BTM sur chaque cluster en identifiant 10 sujets et en conservant le meilleur, car ce modèle ne peut pas rechercher un unique sujet dans un ensemble de données. Ensuite, nous avons conservé les 20 mots clefs les plus pertinents de ce sujet

et extrait des phrases-clefs en tant que combinaisons de ces mots, en ne conservant que celles apparaissant dans au moins 1% des tweets du cluster. Nous avons retenus les 20 plus fréquentes. Ici, la fréquence correspond au nombre de fois où les deux mots-clefs apparaissent dans un même tweet. Nous présentons un échantillon de résultat, pour les garder lisibles, pour un des clusters tiré aléatoirement. Notons que le temps d'exécution de BTM rend cette approche non pertinente pour notre cas d'application. Nous présentons ces résultats, sur le jeu de données tempête Alex, à titre illustratif.

TABLEAU 6.4 – Résultat de simili extraction de phrase-clef basée sur BTM sur un cluster tiré aléatoirement, le 0 comptant 4274 tweets

Mots des sujets	les 20 phrases clefs de taille 2 les plus fréquentes
`vent`, `aller`, `pouvoir`, `faire`, `soir`, `bien`, `chez`, `sans`, `pluie`, `avant`, `nuit`, `demain`, `quand`, `passer`, `quelque`, `beau`, `fois`, `arriver`, `jamais`, `falloir`	`vent pluie`, `vent nuit`, `pouvoir faire`, `vent pouvoir`, `vent soir`, `vent demain`, `aller bien`, `aller faire`, `aller demain`, `soir demain`, `vent quand`, `aller chez`, `nuit demain`, `vent aller`, `vent chez`, `chez demain`, `aller soir`, `aller nuit`, `aller pluie`, `faire chez`

Il n'y a pas de phrase-clef longue de plus de deux mots dans le tableau puisque aucune d'entre elles n'apparaît assez pour être retenue. Cette façon d'identifier des descripteurs avec plus de contexte montre sa limitation. `vent` n'a par exemple pas besoin de plus d'information pour être compris. Or il apparaît dans 8 des 20 phrases retenues. Cela est du au fait que comme vent apparaît beaucoup dans le corpus, il apparaît nécessairement avec une assez grande diversité d'autres mots dans les messages qui le contiennent. Ainsi, vérifier simplement l'apparition dans le même message des mots des sujets pour en faire des phrases clefs peut en fait biaiser l'interprétation de ces mots.

6.3.6.2 Comparaison à BERTopic et à la sélection DF-IDF

Nous comparons nos résultats à ceux obtenus par BERTopic (Grootendorst, 2022) et avec notre score DF-IDF. BERTopic supprime le # et considère un hashtag comme n'importe quel autre mot. L'objectif de cette approche est d'identifier des sujets cohérents sous forme de mots les mieux notés par groupe de documents, sans nécessairement décrire l'ensemble du regroupement. Sur l'ensemble de données de la tempête Alex, il considère que 84% de l'ensemble de données concernent le même sujet (59% de l'ensemble de messages), dans lequel la probabilité du mot le mieux noté `maritime` est de 2,5%, ou qu'ils sont des contenus hors sujet (35% de l'ensemble de messages). Pour les données sur l'Ukraine, elle écarte 49% des messages et 29% sont dans le même sujet dont le meilleur mot `russia` a une probabilité de 4,7%. Cette approche ne semble pas adaptée à la description de regroupements de tweets.

Descripteurs sans UMMs Afin de comparer les trois approches, nous avons configuré BERTopic et la sélection par DF-IDF pour qu'ils produisent 20 descripteurs par cluster. Elles peuvent ainsi identifier un nombre de descripteurs pour la partition similaire à celui obtenu avec la méthode par ablation. Un des avantages de la sélection par ablation est d'identifer plus d'entités nommées, qui sont des éléments importants des descriptions. Nous présentons celles identifiées par les différentes approches dans le tableau 6.5, sans les UMMs car BERTopic procède à sa propre tokenisation. DF-IDF identifie uniquement

87 descripteurs différents pour Alex et 68 pour l'Ukraine, ce qui limite leur diversité et indique que la majorité des mots ayant un score DF-IDF élevé sont communs à de nombreux clusters. Les tableaux 6.5a et 6.5b présentent les entités nommées relevées parmi les descripteurs identifiés par les trois approches.

TABLEAU 6.5 – Descripteurs de cluster faisant référence à des entités nommées, sans unité multi-mot

approche	ablation	DF-IDF	BERTopic
nombre total de descripteurs	200	87	185
Lieux	france, saint-martin-vesubie, bretagne, mercantour, alpha, vésubie, saint-dalmas-de-tende, roya, cordouan, quimper, alpes-maritimes, breil-sur-roya	bretagne, alpes-maritimes, france, morbihan	alpes-maritimes, roya, capbreton, breil, france, bretagne, morbihan, vésubie
Personnes	macron, gerald	/	macron, estrosi,
Organisations	ouestfrance, gendarme (policeman), nice-matin, keraunosobs, meteofrance	franceinfo	anonymousfrance, anonymous, thecointribune, twitter, lesrepublicains
Hashtags	#bretagne, #meteo, #alerte, #morbihan, #mikl	#alpesmaritimes, #bretagne	/

(a) Tempête Alex

approche	ablation	DF-IDF	BERTopic
nombre total de descripteurs	214	68	186
Lieux	kharkiv, russia, kosovo, africa, kyiv	russia,	russia,
Personnes	narendramodi, putin, pmoindia, potus, bbcstever, adamkinzinger, borisjohnson, christogrozev, lavrov, therickwilson, vladimir, biden	biden, putin, potus, trump	putin, trump, potus, biden, abramovich
Organisations	eu_commission, nato	nato	nato, twitter, youtube, netflix, vidio, disney, spotify, wetv, chelseafc
Hashtags	#ukrania, #usa, #standwithukriane, #podcast, #afghanistan, #istandwithukriane, #stoprussia, #putin, #fuckputin	#russia, #stopputin, #putin	/

(b) Guerre Russo-Ukrainienne

Dans ces tableaux, nous pouvons voir que dans les données sur l'Ukraine les entités nommées identifiées par via le DF-IDF ou BERTopic le sont également via l'ablation, en dehors des noms d'entreprises que seul BERTopic a fait ressortir. Dans les données sur la tempête Alex, le constat est similaire : BERTopic n'identifie des entités réellement différentes que pour les organisations. L'identification par les valeurs de I et la distance au centroïde permet une variété plus importante d'entités parmi les tags. Si la description d'un cluster contient l'une ou plusieurs d'entre elles, il est plus simple de raccrocher ce dont traite ses messages à la réalité. Cette variété est donc un avantage pour la description.

Descripteurs avec UMMs Dans les tableaux 6.6a et 6.6b, nous pouvons voir les entités identifiées lorsque les UMMs sont prises en compte.

Nous pouvons voir dans ces résultats que, comme lorsque les UMMs ne sont pas prises

TABLEAU 6.6 – Descripteurs de cluster faisant référence à des entités nommées, avec unité multi-mot

approche	ablation	DF-IDF
nombre total de descripteurs	187	77
Lieux	`alpes-maritime`, `bretagne`, `vesubie roya`, `turquie`, `corse`, `lyon`	`bretagne`, `alpes-maritime`
Personnes	`sarkozy`	/
Organisations	`mediavenir`, `googlenews`, `nice-matin`, `facebook`	`mediavenir`
Hashtags	`#covid19`, `#dunkerque`, `#haiti`	`#alpesmaritime`, `#bretagne`

(a) Tempête Alex

approche	ablation	DF-IDF
nombre total de descripteurs	223	53
Lieux	`russia`, `georgia moldova`, `kenya`, `kyiv tower`, `china`, `canada`, `donetsk`, `no-fly zone`, `poland border`, `residential area`, `france`	`russia`
Personnes	`justintrudeau`, `trump`, `putin`, `rogan`, `avindman`, `zelensky`, `musk`, `poroshenko`, `macron`, `president biden`,	`potus`, `putin`, `biden`, `trump`
Organisations	`nato`, `breitbartnew`, `chipfranklin`, `ukrainian military`	/
Hashtags	`#blacklivesmatter`, `#russia`, `#putin`, `#belarus`, `#trump`, `#smartnew`,	`#stopputin`

(b) Guerre Russo-Ukrainienne

en compte, le score DF-IDF ne permet d'identifier qu'un nombre réduit de descripteurs. Parmi eux, le faible nombre d'entités nommées limitera d'autant plus la facilité avec laquelle il sera possible de lier une description à un évènement réel.

La sélection par ablation montre l'intérêt descriptif des UMMs. L'unité `vesubie roya`, alliant deux noms de rivières relativement éloignées en France, permet avec un seul descripteur de parler d'effets similaires de la tempête dans des endroits différents. L'ampleur de ces effets est reflétée à travers cette association. Si `ukrainian` et `military` ne peuvent être considérés comme des entités nommées séparément, `ukrainian military` en devient une, les mots se spécifiant mutuellement. Il en va de même pour `poland border`.

Ainsi, les UMMs permettent d'identifier une plus grande diversité d'entités nommées, simplifiant au final la compréhension des descriptions.

6.4 Conclusion

Ce chapitre présente la méthode que nous proposons de mettre en place afin de sélectionner des meilleurs descripteurs en mettant à profit le modèle de plongement. Notre objectif est de ne pas avoir recours à des ressources extérieures afin de sélectionner des tags de qualité. La principale limitation des méthodes que nous avons présentées dans le chapitre précédent était de ne considérer que les tokens pour identifier des descripteurs,

les mots seuls pouvant être ambigus ou difficiles à interpréter. Pour contourner ce problème, nous avons choisi de nous appuyer uniquement sur le modèle de plongement que nous employons, Doc2Vec. Ainsi la méthode proposée ne change pas l'architecture que avons présentée au chapitre précédent, et la sémantique capturée par le modèle est au plus proche de nos données.

Comme le modèle apprend à reconnaître les associations de tokens qui font sens étant donné un corpus, nous exploitons les représentations qu'il apprend pour extraire ces associations. En considérant que le modèle apprend les vecteurs de documents en même temps que les mots, les interactions entre les mots et les documents permettent aussi d'isoler les mots qui n'ont pas d'intérêt descriptif. Pour prouver la validité de ces hypothèses, nous avons étudié ces interactions via une étude par ablation.

Les résultats montrent que le modèle peut effectivement être exploité pour identifier des unités de documents s'étendant sur plusieurs mots. Cela permet de donner un sens à des mots qui sont ininterprétables seuls. Cela permet également d'avoir comme descripteurs des collocations spécifiques à l'évènement du corpus ou des entités dont le nom est composé. Ainsi, si les descriptions que nous obtenons avec cette méthode obtiennent de moins bonnes évaluations avec nos métriques, l'intérêt descriptif de cette approche est significatif.

Nos expériences montrent que ce gain descriptif ne peut être exploité en construisant les descriptions via la maximisation de leur DF-IDF, comme avec notre modèle d'assignation en PLNE. On peut avoir recours à une autre méthode, telle que celle que nous présentons dans ce chapitre s'appuyant sur l'impact d'un mot sur son vecteur de document et la distance de ce dernier au centroïde du cluster. Le défaut de cette approche est qu'elle ne contraint pas les descriptions en taille ou en couverture, comme permet de le faire notre modèle de PLNE. Les valeurs de I, qui mesurent l'impact des unités sur leur document, permettent bien d'exclure les mots sans intérêt descriptif pour un cluster. Toutefois, une unité avec une valeur de I non extrême n'a pas nécessairement un grand intérêt descriptif. Une piste d'amélioration de nos travaux est donc de trouver comment concilier notre modèle d'assignation avec les UMMs et les valeurs de I des unités considérées. Les descriptions bénéficieraient ainsi de la qualité induite par leurs contraintes de composition et de l'intérêt descriptif des UMMs, sans pouvoir compter de tags aberrants.

À travers ce chapitre et le précédent, nous avons éprouvé nos différentes hypothèses de travail et identifié comment exploiter plongement et clustering pour décrire des données. Nous avons jusqu'ici travaillé sur des jeux de données statiques. Le prochain problème à aborder est celui de la gestion des données en flux.

Chapitre 7

Vers une description incrémentale

Sommaire

Les travaux présentés dans les chapitres 5 et 6 nous ont permis d'éprouver nos hypothèses de travail et de concevoir des outils permettant de traiter et de décrire des jeux de données textuels figés. Le modèle de plongement Doc2Vec est adapté aux jeux de données que nous traitons et permet d'en extraire de l'information en le combinant à des méthodologies de clustering classiques comme le K-Means. Nous avons vu en outre comment l'exploiter pour identifier des unités multi-mots spécifiques à un corpus, qui présentent un intérêt notable pour la tâche de description de données.

Dans ce chapitre, nous présentons les travaux que nous avons entamés pour adapter notre approche aux données en flux. Les évolutions doivent permettre de s'adapter aux caractéristiques d'un corpus dynamique. La première difficulté est que naturellement, toutes les données ne sont pas connues à un instant donnée du processus. La méthodologie de partitionnement doit donc pouvoir fonctionner sans connaître tout le corpus. L'autre principale difficulté vient des attributs des documents à traiter. Comme les instances

traitées sont des textes, leurs attributs sont leurs mots et chaque nouvelle instance peut contenir un attribut inconnu jusqu'alors. La représentation du document ne peut alors pas se baser sur cet élément, qui n'est pas encore encodé dans les dimensions de l'espace de plongement. Comme Doc2Vec est peu profond et rapide à ré-entraîner, il peut être utilisé dans le contexte incrémental.

Les modifications à apporter sont ainsi les suivantes. D'abord le modèle de plongement doit être tenu à jour, car les données à partir desquelles il capture sa sémantique s'enrichissent constamment. Ensuite, la méthodologie de partitionnement doit évoluer pour pouvoir incorporer des données en continu. Les nouveaux messages doivent être intégrés à la partition courante pour que l'on puisse les rapprocher de ceux qui leur ressemblent ou identifier un nouveau groupe le cas échéant, le tout sans compromettre l'intégrité de la partition. Finalement, les descriptions des groupements doivent être tenues à jour. Un descripteur peut être adéquat à un moment donné et pas à un autre en fonction des messages qu'il décrit.

Nous proposons ici nos expérimentations et résultats exploratoires pour une description de données opérée par un système incrémental.

7.1 Approche incrémentale

Pour pouvoir identifier des groupements de similarités des données reçues en flux, nous nous inspirons de l'approche incrémentale proposée par (Hasan et al., 2019), présentée dans la section 2.3.3. L'approche que nous envisageons suit le comportement général présenté dans l'algorithme 14. Celui-ci décrit les principes de la procédure et présente les problématiques qu'elle soulève. Les détails de l'implantation sont donnés dans la section suivante.

Même si les traitements à appliquer à chaque tweet ne sont pas nécessairement lourds, il est plus approprié de traiter les données reçues en flux par paquets, ou batchs. Cela permet d'effectuer les opérations les plus lourdes pour un batch plutôt que pour chaque tweet, limitant ainsi le coût computationnel.

Les trois premières lignes de l'algorithme 14 correspondent au lancement de la description incrémentale. Comme il faut un premier jeu de données pour pouvoir entraîner le modèle de plongement, il faut en disposer ou le constituer comme on peut le voir ligne 1. Ce corpus D permet d'entraîner le modèle de plongement pour en obtenir l'état initial, ligne 2, qui va être utilisé pour lancer l'écoute du flux S. L'approche doit également disposer d'une partition à partir de laquelle travailler, il faut donc également construire une partition initiale, ligne 3. À ce stade, il ne devrait pas y avoir un grand nombre de sous-évènements dans les données, et donc de clusters dans la partition.

Une fois ces trois éléments à disposition, on peut commencer à écouter le flux pour constituer les batchs à traiter. La limite de constitution des batchs peut être une durée, comme c'est le cas dans l'algorithme, ou un volume de batch. Les deux approches peuvent être inter-changées. Une fois sa limite atteinte, le batch est traité pour que son contenu mette à jour la partition et ses descriptions si nécessaire. Pour ne pas interrompre l'écoute du flux et potentiellement ainsi rater des données, les processus de l'écoute du flux et du traitement des batchs doivent être effectués en parallèle.

Le modèle de plongement doit être à jour de l'utilisation de la langue sur le flux écouté pour que les représentations des nouveaux tweets soient pertinentes. Cela peut se

Algorithm 14: Principe de la description incrémentale

input : stream Twitter S, nombre de clusters initiaux k, seuil de ré-entraînement temporel *temp*, durée de collecte pour constituer un batch z, seuil de distance au centroïde λ, seuil de ré-entraînement *retrain*

```
1  requêter le jeu de donné initial D
2  entraîner le modèle de plongement model sur D
3  calculer une partition initiale P à k clusters
4  construire l'ensemble de description E_P pour P
5  batch de tweets B ← [ ]
6  date de début z_B ← date à l'instant courant /* z_B est le top temporel du
       début de la constitution de B                                    */
7  Tant que nouveau tweet t dans S faire
8      pré-traiter t
9      ajouter t à D
10     ajouter t à B
11     Si la durée de constitution de B ≥ z alors
12         traiter_batch(B, D, model, P, E_P, λ, retrain)
13         z_B ← date à l'instant courant
14         B ← [ ]
15     Si durée depuis le dernier entraînement de model ≥ temp alors
           reinitialiser(D, model, P, E_P);
```

mesurer au moment de traiter le contenu des messages, quand un batch est traité comme nous allons le voir dans l'algorithme 16. Toutefois, des variations subtiles du langage des données traitées, qui peuvent échapper au suivi basé sur les mots opéré pendant le traitement de B, s'accumulent au fil du temps. Il faut donc également ré-entraîner le modèle régulièrement, tel qu'en ligne 15. Cette mise à jour déclenche d'autres procédures, comme nous le voyons dans l'algorithme 15.

Les seuils λ et *retrain* visibles dans l'algorithme 14 sont utilisés dans l'algorithme 16, qui présente le comportement que doit suivre le traitement d'un batch.

Algorithm 15: Principe de la réinitialisation : *reinitialiser*

input : corpus D, modèle de plongement *model*, partition $\mathcal{P}$, ensemble de description $E_{\mathcal{P}}$

```
1  ré-entraîner model sur D
2  recalculer P
3  reconstruire E_P
4  renvoyer model, P et E_P
```

L'algorithme 15 présente le comportement à suivre lorsque le modèle de plongement est ré-entraîné. Changer le modèle de plongement change l'information contenue dans chacune des dimensions des espaces de représentation de mots et de documents. Même si les vecteurs sont de mêmes dimensions, mesurer la distance entre ceux construits par le modèle initial et celui ré-entraîné n'a aucun sens. Il faut donc recalculer la partition

$\mathcal{P}$ dans ce nouvel espace. Pour cela, on peut s'appuyer sur son dernier état connu pour avoir une indication du nombre de cluster à trouver avant de relancer l'écoute du flux. Finalement, comme $\mathcal{P}$ est recalculée, il faut également mettre à jour les descriptions $E_{\mathcal{P}}$.

Algorithm 16: Principe du traitement d'un batch : *traiter_batch*

input : batch de tweet B, corpus D, modèle de plongement *model*, partition $\mathcal{P}$, ensemble de description $E_{\mathcal{P}}$, seuil de distance au centroïde λ, seuil de ré-entraînement *retrain*

```
1  trace_retrain ← 0
2  maj_desc ← tableau associatif vide
3  Pour chaque tweet t dans B faire
4      Si DF-IDF(mots de t, D) << ou >> DF-IDF(mots de t, D \ B) alors
         trace_retrain+ = 1;
5      inférer t⃗ avec model /* le modèle renvoie aussi les impacts des mots
           t sur t⃗                                                    */
6      candidats ← [ ] Pour chaque cluster C dans P faire
7          c ← centroïde de C
8          Si distance(t⃗, c) ≤ λ alors ajouter c à candidats;
9      Si candidats ≠ ∅ alors
10         assigner t au cluster C dont le centroïde c est le plus proche de t⃗
11         ajouter les mots de t avec leur impact sur t⃗ à maj_desc[C]
12         Si |candidats| ≥ 2 alors dé-fragmenter P;
13     sinon mettre à jour P pour intégrer t;
14 Si trace_retrain ≥ retrain alors reinitialiser(D, model, P, E_P);
15 mettre à jour E_P en fonction de maj_desc
```

La procédure de traitement de batch, dans l'algorithme 16, donne les grandes lignes du comportement incrémental que nous souhaitons mettre en place pour traiter les données en flux. Lorsqu'un nouveau tweet t est traité, il est comparé à chaque centroïde de cluster existant. S'il en existe un suffisamment proche, c'est à dire plus proche que le seuil λ, alors t lui est raccroché tel que l'on peut le voir ligne 10. Avec un modèle de plongement à jour, cela permet de traiter les nouveaux messages proches de ceux que l'on connaît déjà.

Il y a toutefois plusieurs autres points de difficulté à aborder dans le cadre incrémental, regroupés dans la procédure décrite dans l'algorithme 16. Notons que chacune des pistes présentées ici pour aborder ces points est explorée dans ces expérimentations mais reste encore à discuter.

La première est le comportement à mettre en place pour traiter les tweets qui ne sont proches d'aucun des clusters faisant déjà partie de la partition. Lorsqu'aucun cluster n'est assez proche de t, il faut en créer un nouveau pour intégrer ce dernier à $\mathcal{P}$. Cette modification doit être opérée en garantissant l'intégrité et la cohérence globales de la partition. Elle peut donc amener d'autres modifications. Cette étape du processus est présentée en section 7.2.4.

Le suivi de l'utilisation du vocabulaire des données s'opère ligne 4. Nous souhaitons le baser sur le DF-IDF, que nous avons présenté en section 5.1.2.2. La question de son

utilisation est toutefois encore ouverte. Celle que nous avons testée est détaillée en section 7.2.3.

Une problématique centrale du traitement des données en flux via leur partitionnement est la fragmentation. Plusieurs clusters peuvent se former séparément alors qu'ils ne devraient en former qu'un, du fait de l'ordre dans lequel les données sont reçues. Une façon de détecter cette situation est de relever quand un tweet est assez proche de plusieurs clusters pour qu'il soit acceptable de le raccrocher à chacun d'entre eux, comme on peut le voir ligne 12. La procédure que nous avons expérimentée est présentée en section 7.2.5.

Finalement, les descriptions des données doivent être tenues à jour au fil du traitement des nouveaux messages. Nous avons vu dans le chapitre 6 que l'impact d'un mot sur le vecteur de son document permet d'identifier si ce mot à un intérêt descriptif pour un cluster. Cet impact correspond en fait à la mise à jour du vecteur de document calculée par rapport à ce mot par le modèle. Celle-ci peut être récupérée à l'inférence d'un vecteur. En associant cette information à la distance entre les documents et le centroïde de leur cluster, les descriptions peuvent être construites au fil de la réception des données. Ce processus est décrit en section 7.2.6.

7.2 Architecture proposée

La figure 7.1 présente l'architecture du prototype de description incrémentale que nous avons expérimenté. Nous détaillons chacun de ses composants et étapes clefs dans les sections suivantes. Cette approche soulève notamment des problématiques de seuil que nous détaillons dans les sections correspondantes.

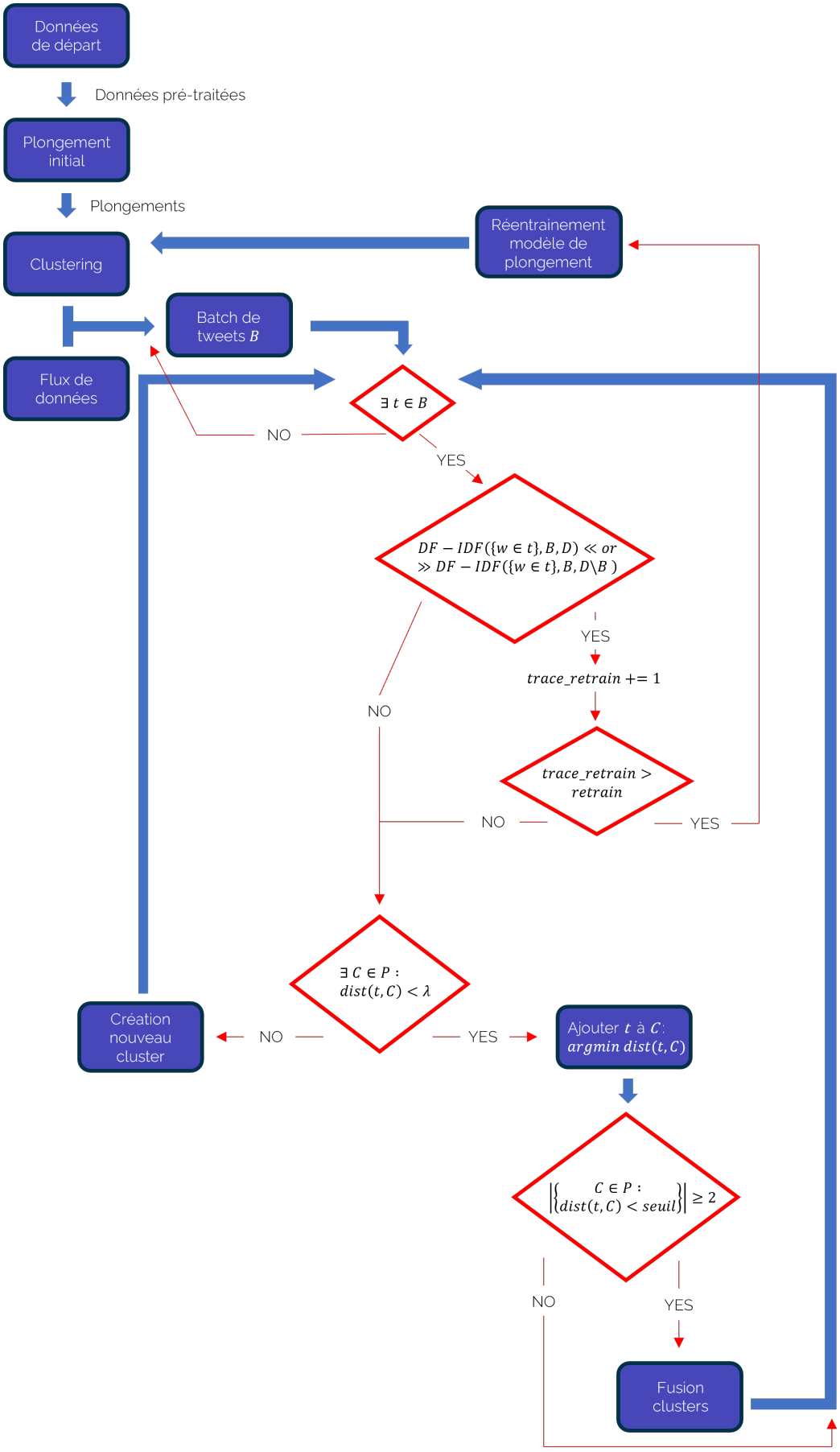


FIGURE 7.1 – Architecture de description incrémentale

7.2.1 Initialisation

Il faut des ressources initiales pour lancer la description incrémentale. D'une part, un modèle de plongement pour inférer les vecteurs des tweets entrants, et d'autre part un clustering initial. Dans ce but, nous proposons d'utiliser la même requête que celle qui va permettre d'écouter l'évènement cible par la suite. Ainsi, on peut construire un premier jeu de données et lui appliquer les mêmes pré-traitements que nous avons décrit dans le chapitre 4.

Dans les expérimentations que nous avons menées, nous avons ordonnés les corpus sur la tempête Alex et sur la guerre Russo-Ukrainienne en fonction de la date de publication des tweets. Leur première moitié a été utilisée pour l'entraînement initial du modèle de plongement et l'autre pour simuler un flux de tweet. Ce corpus initial permet d'entraîner le modèle de plongement de départ et pour y identifier la partition initiale, nous proposons d'utiliser K-Means. Nous l'avons exécuté avec $K = 5$, paramètre choisi arbitrairement.

7.2.2 Collecte des données et batch

La gestion des flux de données est souvent effectuée par batch, c'est-à-dire par lot ou paquet de données entrantes. Dans un cas d'application comme celui que nous traitons, on pourrait envisager de traiter les tweets un à un. Toutefois, cela poserait rapidement un problème de coût computationnel. En effet, en fonctionnant tweet à tweet, les opérations de mise à jour générale de la partition serait aussi nombreuses que le flux produit de messages.

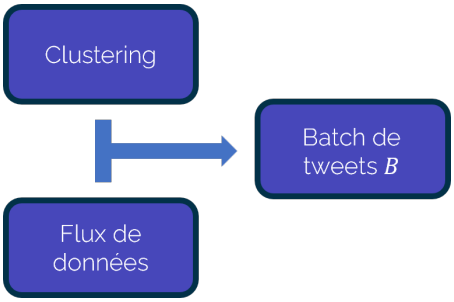


FIGURE 7.2 – Zoom architecture : constitution des batchs de tweets

On peut considérer deux approches pour arrêter la constitution des batchs de données. Dans le premier cas, la méthode consiste à collecter un certain volume de données pour constituer un lot. Une fois cette limite atteinte, le batch est traité et le prochain commence à être agrégé. La seconde méthode consiste à définir une durée pendant laquelle le batch est constitué. Le flux est écouté et à chaque top de la limite de temps définie, un batch est envoyé au traitement.

Les deux manières conviennent pour les flux de données réguliers, dans lesquels il n'y aura pas de changement important du volume de données issu du flux au cours du temps. Les problèmes suivants peuvent se poser dans un flux irrégulier. Une limite de temps peut amener à lancer le traitement d'un énorme ou au contraire très petit volume de messages. Une limite de volume peut décupler le nombre d'appel au traitement de batch ou au contraire le limiter fortement. Comme nous travaillons sur un flux Twitter, la quantité de

messages reçue varie au cours du temps, selon l'actualité de la cible écoutée. Il nous faut donc adapter le critère d'arrêt de constitution des batchs.

Pour ce faire, nous proposons d'écouter le flux pour constituer un premier batch B pendant une durée paramétrée t_B, et de mesurer le nombre de tweets collectés n_B pendant ce temps. La durée de collecte des batchs suivants dépend du volume de tweets entrant. Si on collecte à nouveau n_B tweets pendant une durée proche de t_B, alors la durée de collecte reste constante. Lorsque le flux augmente, il faut s'adapter à l'augmentation de charge en réduisant la durée t_B. Ainsi, si le volume de tweets n_B est atteint en un temps t'_b significativement plus court que t_B, t_B devient t'_b pour le batch suivant. Cette durée est maintenue tant qu'elle permet de collecter n_B tweets, et itérativement réduite de la même façon le cas échéant.

À l'inverse, si au bout de la durée t_b on a toujours pas collecté n_B tweets, le batch est envoyé au traitement et la durée de collecte du prochain batch est rallongée. Cela peut être répété jusqu'à revenir à la durée paramétrée initiale. Ainsi, on s'adapte aux variations du volume de données sortant de Twitter.

Notons que c'est lors de la constitution des batchs que les pré-traitements sont appliqués aux tweets. L'état de la partition est figé pendant le traitement d'un batch. Ainsi, chaque lot de messages est traité en tenant compte du résultat du traitement du dernier batch.

7.2.3 Gestion du modèle de plongement

Maintenir le modèle de plongement des documents et des mots à jour est crucial. Le clustering incrémental sera en effet construit sur les vecteurs de documents que celui-ci infère. Il doit donc être à jour pour représenter efficacement la sémantique des nouveaux tweets et pour que la distance entre les documents permettent d'identifier les sous-évènements.

Doc2Vec s'appuie sur la sémantique distributionnelle pour apprendre les interactions entre les documents, dont les unités de base sont les mots. C'est à travers les mots que le modèle saisit les interactions mot-contexte. Il ne peut naturellement pas s'appuyer sur des mots ou sur des interactions qui n'apparaissent pas dans les données d'entraînement. Il faut donc pouvoir identifier les nouveaux mots et les nouveaux usages des mots connus. Nous souhaitons le faire d'une manière indépendante du modèle, pour s'appuyer directement sur les données brutes.

Pour cela, nous testons la capacité du DF-IDF à suivre ces changements. Sa formule est rappelée dans l'équation 7.1. Nous l'adaptons au contexte incrémental pour devenir l'équation 7.2.

$$\text{DF-IDF}(w, \mathcal{C}, \mathcal{P}) = \frac{N_w(\mathcal{C})}{|\mathcal{C}|} \log(\frac{N(\mathcal{P})}{N_w(\mathcal{P})}) \tag{7.1}$$

$$\text{DF-IDF}(w, B, D) = \frac{N_w(B)}{|B|} \log(\frac{N(D)}{N_w(D)+1}) \tag{7.2}$$

La différence entre les formulations permet de calculer le score d'un mot dans un batch par rapport à l'état courant du corpus plutôt que dans un cluster par rapport à une partition. La partie IDF est également modifiée pour ne pas opérer de division par 0.

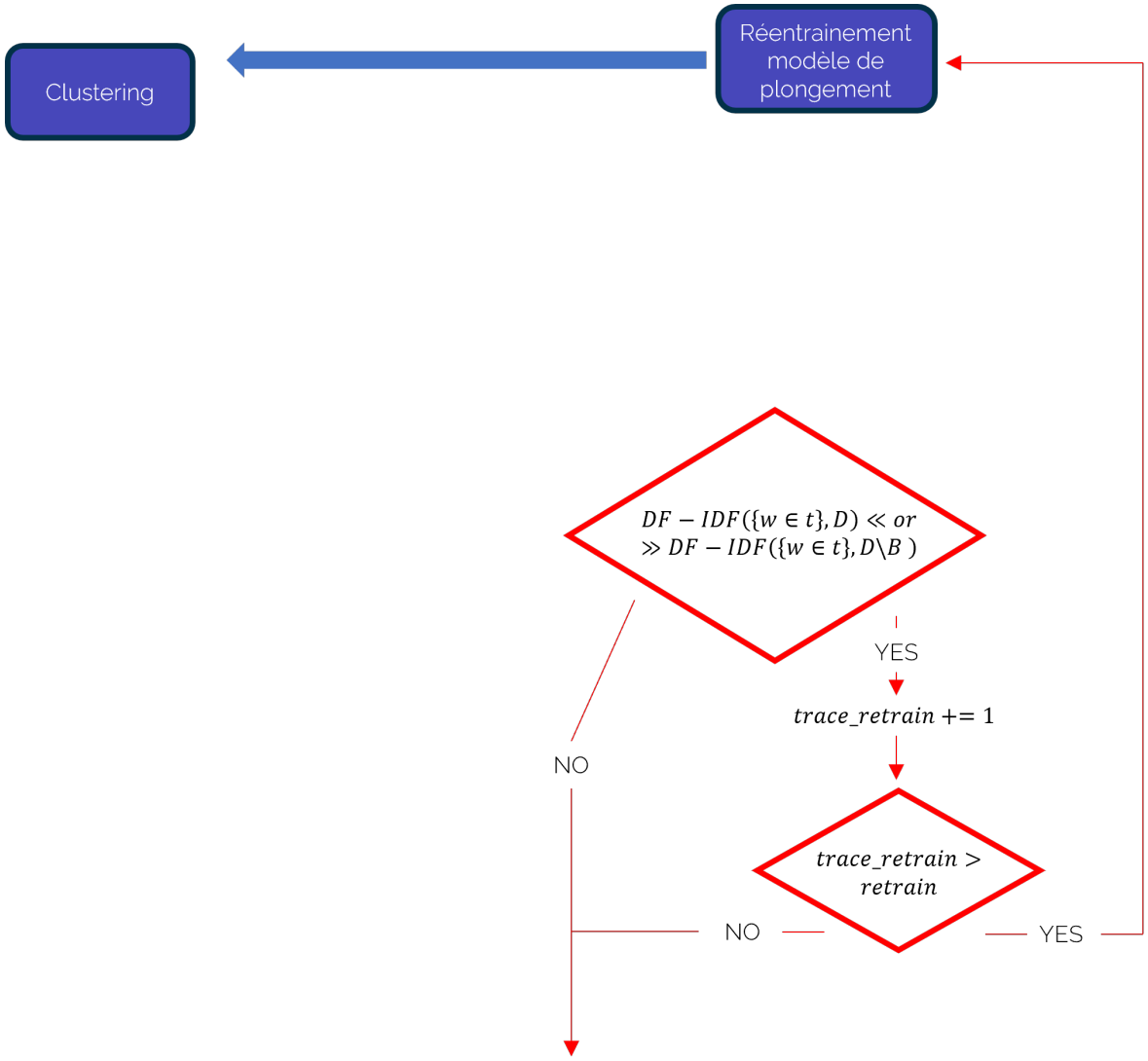


FIGURE 7.3 – Zoom architecture : gestion du modèle de plongement

Dans les travaux exploratoires que nous avons menés, nous comparons $DF-IDF(w, B, D)$ et $DF-IDF(w, B, D \setminus B)$. Si la prise en compte du batch pour calculer le score d'un mot change ce score significativement, il est probable que cela traduise un changement de son utilisation. Il faut alors augmenter le nombre de cas identifiés dans ce batch justifiant de ré-entraîner le modèle de plongement. Lorsque que ce décompte dépasse le seuil $retrain$, le modèle est mis à jour.

Le top temporel, indépendant du suivi avec le DF-IDF est définit à un jour. Le modèle est ré-entraîné à la fin de chaque journée de collecte.

Réentraîner le modèle rend obsolète la partition courante, il faut donc la recalculer. Cette étape est toutefois simplifiée du fait que les documents à re-partitionner sont connus. Il est donc possible d'utiliser un algorithme de clustering classique, paramétré avec le dernier nombre de clusters connu. C'est l'information la plus précise dont on dispose sur la topographie des données au moment où le modèle de plongement doit être ré-entraîné. S'il n'est pas garanti que ce soit également le nombre de cluster à identifier dans le nouvel espace de représentation, c'est le meilleur a priori dont on dispose.

Une fois la partition recalculée, on peut recommencer à traiter les batchs de tweets entrants.

7.2.4 Positionnement dans la partition et création de nouveaux clusters

Pour découvrir le nombre de clusters, il faut en créer de nouveaux lorsqu'il est nécessaire de le faire. Ce processus s'appuie nécessairement sur les données entrantes et sur l'état de la partition au moment de leur réception.

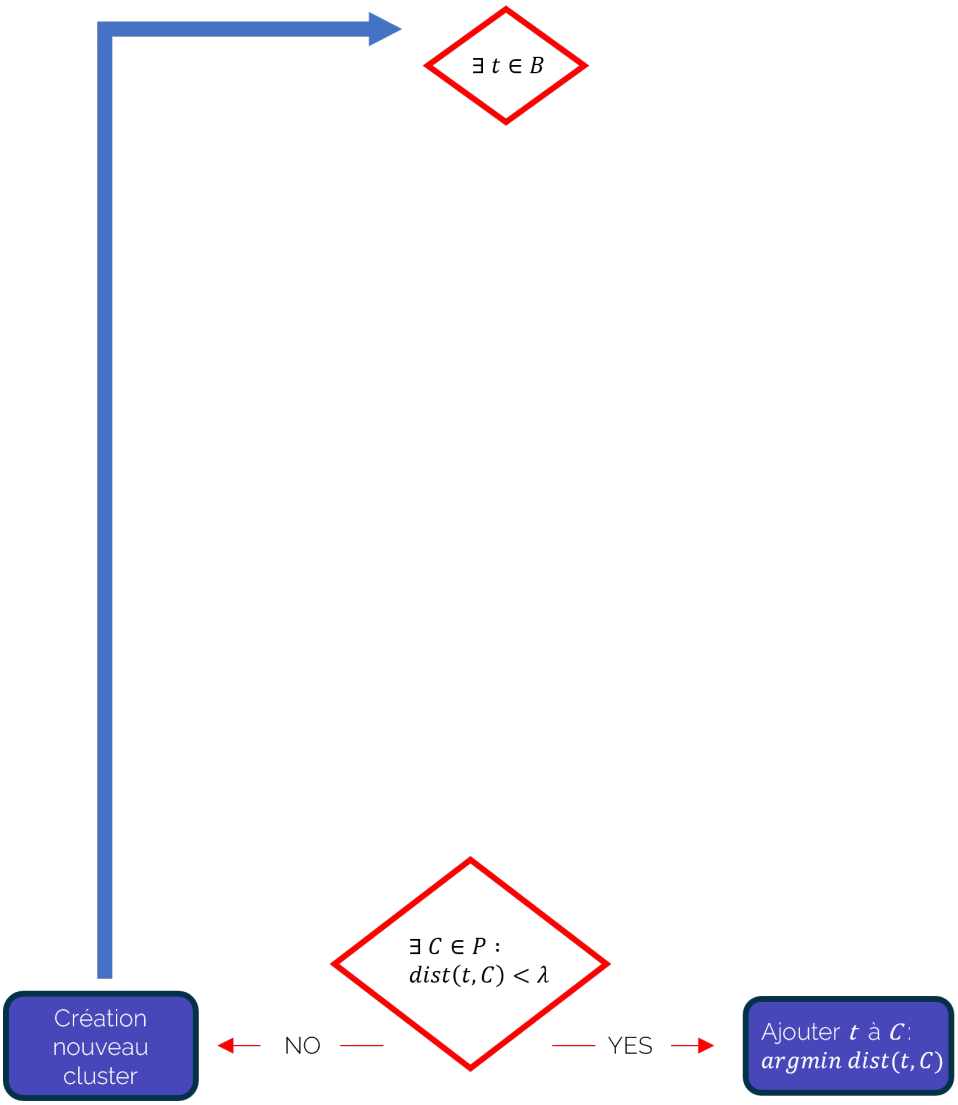


FIGURE 7.4 – Zoom architecture : initialisation de nouveaux clusters

L'approche que nous souhaitons mettre en place est inspirée de celle de (Becker et al., 2011). Si un nouveau document est suffisamment proche d'un des clusters existants, on le lui rattache. On vérifie à travers ce test s'il existe dans la partition une structure qui puisse contenir ce nouvel élément.

Si ce n'est pas le cas, c'est que ce nouveau document a des caractéristiques trop

éloignées de celles qui sont déjà connus. Il devient donc le centroïde d'un nouveau cluster, qui permettra d'y raccrocher les prochains messages similaires.

Cette méthode nécessite de fixer la distance à partir de laquelle on considère qu'un document entrant est trop loin des clusters existants pour y être raccroché. La valeur de cette distance, le seuil paramétré λ, n'est pas humainement interprétable. Elle se mesure dans l'espace de plongement et représente la différence sémantique entre les documents. On ne peut pas attribuer une signification à sa valeur absolue. La seule façon de l'exploiter est donc de comparer les distances au sein de cet espace. Il convient ainsi de fixer ce seuil en examinant les statistiques des clusters. Comme nous évaluons les messages en batchs, nous proposons d'évaluer la moyenne de la distance au centroïde dans les clusters et de fixer le seuil λ en fonction de ces moyennes.

Ajouter des clusters à la volée en fonction des instances entrantes en s'appuyant sur un seuil de distance soulève une autre problématique. Nous considérons que les documents doivent être attribués au cluster dont ils sont les plus proches. Ainsi, la création d'un nouveau cluster doit s'accompagner de la restructuration du clustering existant. Celle-ci s'opère simplement, une fois le nouveau cluster créé. On rattache au nouveau cluster tous les documents plus proches de lui que du centroïde de leur cluster actuel. Procéder de la sorte à la création de chaque nouveau cluster devrait maintenir l'intégrité de la partition.

Dans ces expérimentations, nous considérons que la partition est fragmentée s'il y a plusieurs clusters pour lesquels la distance entre le centroïde et un nouveau tweet est inférieure à λ. Il faut corriger ce problème pour rétablir l'intégrité de la partition.

7.2.5 Défragmentation : fusion de clusters similaires

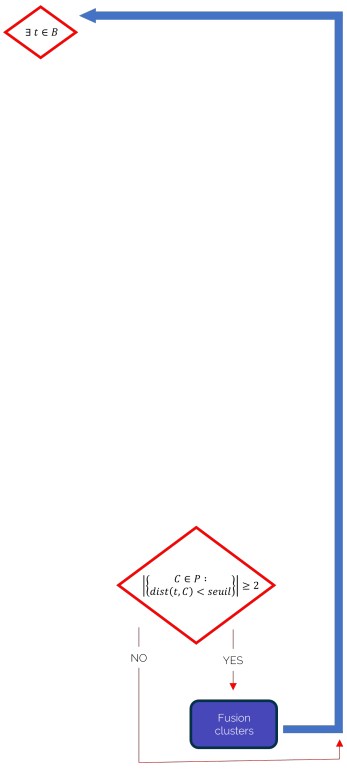


FIGURE 7.5 – Zoom architecture : défragmentation de la partition

La fragmentation des clusters d'une partition est un problème qui se pose lorsque leur constitution n'a pas pu prendre en compte toutes les données. De nouvelles instances peuvent ainsi être isolées ou même former un nouveau groupe alors qu'elles auraient été intégrées à la partition autrement si elles avaient été connues au moment du clustering initial. Ce problème est complexe à aborder. Sa détection s'opère simplement : si une instance pourrait être attribuée à plusieurs clusters selon leur critère de formation, cela révèle un problème dans la partition. Procéder à la défragmentation est toutefois complexe.

Comme ces travaux constituent une première approche du problème, nous l'abordons de la même façon que (Becker et al., 2011). Tous les clusters auxquels on peut attribuer une même instance sont fusionnés. Cela devrait permettre de maintenir l'intégrité de la partition. Le parti pris de cette approche est fort, puisqu'une seule instance commune suffit à déclencher la fusion.

7.2.6 Construction des descriptions

L'impact des mots sur le vecteur de leur document est mesurable dans les coordonnées des vecteurs, nous l'avons vu dans le chapitre 6. Pour pouvoir accéder à cette information sans avoir de calcul supplémentaire à effecteur après l'inférence du document, nous avons modifié l'implantation de cette dernière. Elle procède ainsi exactement de la même manière que le modèle original, mais permet de récupérer l'impact direct d'un mot sur la position de son document.

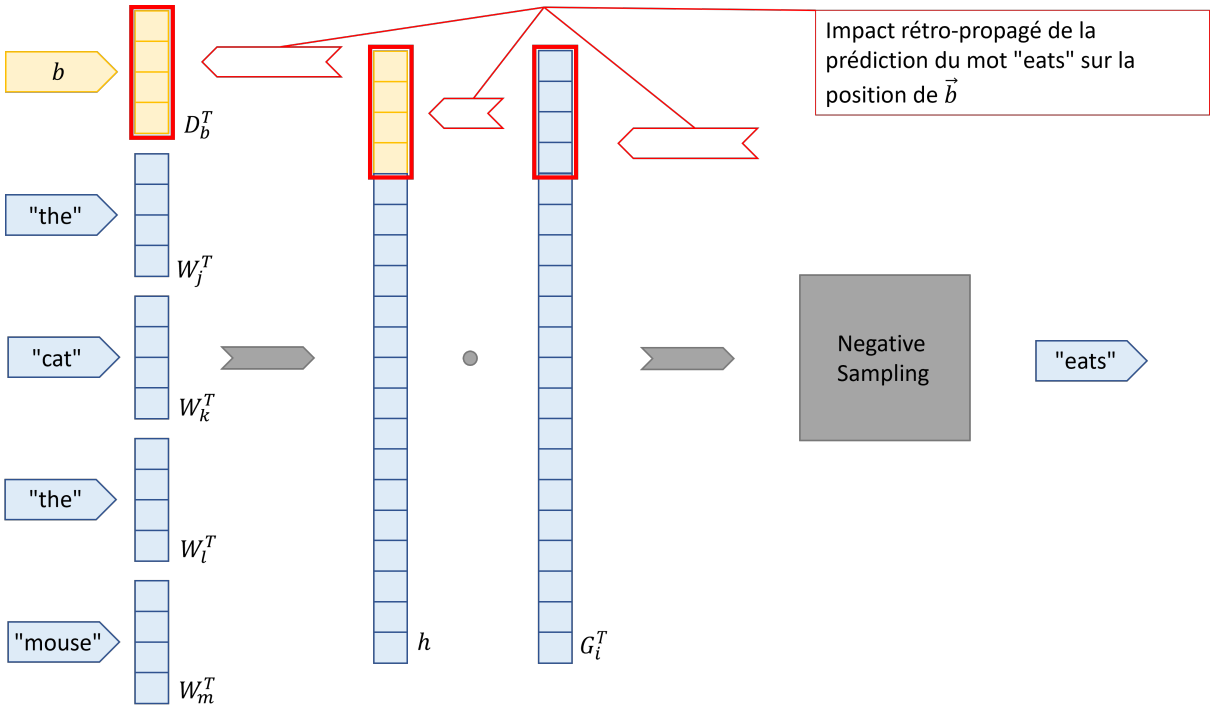


FIGURE 7.6 – Capture de l'impact direct de la présence du mot "eats" dans le document b

La figure 7.6 présente le principe de récupération de l'impact d'un mot directement à l'inférence d'un vecteur. Rappelons que pour Doc2Vec, l'inférence d'un vecteur d'un nouveau document consiste à apprendre ce vecteur de la même manière que ceux des

documents connus. À l'inférence, les représentations des mots sont celles déjà apprises par le modèle et elles ne sont pas mises à jours.

Ainsi l'impact d'un mot sur son vecteur de document peut se mesurer en sommant les modifications rétro-propagées vers le document lorsque ce mot est la cible de la prédiction. La modification d'implantation consiste simplement à retenir cette somme pour chaque mot du document, et toutes les renvoyer avec le vecteur inféré. On peut alors mesurer si ce mot rapproche ou éloigne son document de son cluster, en soustrayant son impact au vecteur de son document et en comparant le résultat au centroïde du cluster.

Soit le tweet d contenant le mot w. Notons $u(w, d)$ la somme des modifications apportées à $\vec{d}$ du fait des prédictions de w lors de l'inférence de $\vec{d}$. Si d ne contenait pas w, sa position que nous notons $\vec{d'}$ serait $\vec{d} - u(w, d)$. Notons que $u(w, d)$ est un vecteur de la même dimension que $\vec{d}$. Il est ainsi possible de mesurer l'impact de w sur $\vec{d}$ sans avoir à inférer de nouveau $\vec{d}$ en masquant w comme nous l'avons fait dans le chapitre 6. On peut alors établir le score I' pour identifier les impacts des mots sur leur document par rapport à leur cluster en temps linéaire par rapport au nombre de documents à traiter. Ce score se calcule avec :

$$I'(w, d, c) = (dEUC(c, \vec{d}) - dEUC(c, \vec{d'}))^2 \tag{7.3}$$

où c est le centroïde du cluster traité.

Étant donné un cluster $\mathbf{C}$ de centroïde c, la description est faite des mots des γ documents les plus proches de c ayant une valeur I' non extrême pour $\mathbf{C}$.

7.3 Expérimentations

Nous avons mené plusieurs séries d'expérimentations pour tester les composants et les procédures de description incrémentale que nous proposons.

7.3.1 Partition initiale

Dans les expérimentations menées, nous distinguons des profils pour la partition initiale obtenue avec K-Means.

Les tableaux de la figure 7.7 montrent les profils que nous avons pu identifier pour le clustering initial de la description incrémentale. De la même manière que dans les chapitres précédents, nous avons répété K-Means plusieurs centaines de fois avec $K = 5$ sur la première moitié de nos deux jeux de données pour évaluer la stabilité de la partition de départ.

De cette manière, nous avons identifié un profil stable pour les données sur la tempête, présenté dans le tableau 7.7a. Nous pouvons voir qu'il y a un cluster qui agrège plus de 70% des données partitionnées, un cluster de 1500 tweets et trois autres plus petits et de taille similaire.

Dans le second corpus, celui sur la guerre en Ukraine, le clustering initial semble pouvoir suivre deux modèles de topographie. Le premier, apparaissant dans environ les deux tiers des cas, est celui présenté dans le tableau 7.7b. Plus de 70% des données sont réparties entre deux clusters, un de 4000 et l'autre de 3000 tweets. Un autre se compte en milliers et un en centaines. Le second modèle de topographie comporte un cluster

numéro du cluster	taille en tweet
0	478
1	1508
2	257
3	7443
4	322

(a) Données tempête Alex

numéro du cluster	taille en tweet
0	1230
1	3276
2	5
3	588
4	4057

(b) Données tempête guerre Russo-Ukrainienne, premier profil

numéro du cluster	taille en tweet
0	6088
1	225
2	562
3	5
4	2276

(c) Données tempête guerre Russo-Ukrainienne, second premier profil

FIGURE 7.7 – Profils types du clustering initial, obtenu avec un K-Means où $K = 5$ sur la moitié du jeu de donnée

rassemblant plus de 60% des messages, avec un autre cluster comptabilisant en 20 et 23% des données. Le plus grand cluster est moitié plus grand que dans le premier profil, tous les autres sont plus petits. Cette situation a lieu dans le dernier tiers des cas. Les deux profils comportent toujours un cluster très petit, qui contient moins de 10 tweets.

Il est difficile d'estimer l'impact de l'état de la partition initiale sur les résultats finaux, puisqu'elle est souvent retravaillée. À paramétrage égal toutefois, nous constatons, comme nous allons le voir dans la section suivante, que les résultats une fois toutes les données ingérées sont presque les mêmes. Cette observation tend à indiquer que le résultat de l'approche incrémentale est moins dépendant de son initialisation que de la procédure de traitement du flux de données.

7.3.2 Stabilité des résultats finaux

Toutes les exécutions de description incrémentale menées avec les mêmes seuils sur les données aboutissent presque aux mêmes partitions. Concrètement, si $\lambda < 0.08$, on retrouvera entre un quart et la moitié des données dans des clusters ne comptant qu'un seul tweet. Ensuite, autour de 200 clusters seront d'une taille se chiffrant en centaines et un dernier cluster comportera plusieurs milliers de messages.

Cette stabilité est inattendue. En effet dans ce prototype, K-Means est appelé à chaque

ré-entraînement du modèle de plongement, ce qui implique autant d'initialisations K-Means++ semi-aléatoires. De plus, si on compte plusieurs milliers de clusters avec un seul tweet, alors on compte au moins autant de procédures d'initialisation de nouveau cluster, décrites dans la section 7.2.4. On pourrait donc s'attendre à ce que ces processus amplifient l'aléa induit par les K-Means++, mais cela ne semble pas être le cas.

Il semblerait donc que la structuration sous-jacente des données telles que représentées avec Doc2Vec soit identifiable de manière robuste, en choisissant un seuil λ adéquat. Cette observation est à nuancer en gardant en considération le grand nombre de tweets isolés qui sont détectés avec $\lambda \leq 0,008$.

7.3.3 Topographie de la partition

Des différents seuils que notre prototype comporte, celui qui a un effet le plus visible sur ses résultats est λ, celui dont dépend l'attribution d'un tweet à un cluster.

Valeur de λ	nombre total de clusters	$1 \leq taille < 10$	$10 \leq taille < 100$	$100 \leq taille < 1000$	$1000 \leq taille$
0,01	9945	9705	223	17	0
0,02	9245	9011	233	1	1 (4146)
0,03	8299	8074	224	0	1 (5854)
0,05	6404	6173	223	0	1 (8431)
0,08	4046	3916	129	0	1 (12333)
0,1	3087	2935	95	0	1 (14101)
0,2	566	537	28	0	1 (18444)
0,3	217	208	8	0	1 (19574)
0,5	95	91	3	0	1 (19859)
0,6	74	72	1	0	1 (19896)
0,7	41	40	0	0	1 (19961)
0,8	35	34	0	0	1 (19964)

(a) Topographie de la partition obtenue avec la description incrémentale dans les données sur la tempête Alex

Valeur de λ	nombre total de clusters	$1 \leq taille < 10$	$10 \leq taille < 100$	$100 \leq taille < 1000$	$1000 \leq taille$
0,01	9138	8825	304	9	0
0,02	8929	8622	298	9	0
0,03	8851	8559	285	6	1 (1614)
0,05	8621	8321	298	1	1 (2418)
0,08	8181	7908	271	1	1 (3520)
0,1	7725	7499	225	0	1 (5084)
0,2	5580	5507	68	3	2 (6437, 3007)
0,3	3122	3081	37	2	2 (8137, 5033)
0,5	886	872	10	2	2 (13947, 2473)
0,6	609	599	6	1	3 (14012, 2079, 1153)
0,7	446	431	11	2	2 (13475, 3253)
0,8	340	325	13	1	1 (17109)

(b) Topographie de la partition obtenue avec la description incrémentale dans les données sur la guerre Russo-Ukrainienne

FIGURE 7.8 – Nombre et taille des clusters obtenus en fonction de la valeur de seuil λ

Comme on peut s'y attendre, plus la valeur de λ est grande, moins il y a de clusters dans la partition et réciproquement. Dans le tableau 7.8a, notons qu'à partir de $\lambda = 0,1$, près des trois quarts de toutes les données sur la tempête Alex sont agrégées dans le même ensemble. À partir de cette valeur, la distance moyenne au sein du grand cluster

est supérieure à 0,06. La tendance est un peu différente dans les données sur la guerre en Ukraine. C'est à partir de $\lambda = 0,5$ qu'on trouve un cluster qui couvre presque les trois quarts du corpus. Mais nous pouvons toutefois voir dans le tableau 7.8b que même pour ces valeurs de seuil élevées, sauf pour 0,8, on trouve encore plusieurs clusters dont la taille se compte en milliers.

Une autre différence d'observation entre les jeux de données est que l'on trouve plusieurs milliers de tweets isolés jusqu'à $\lambda = 0,3$ côté Ukraine, mais seulement jusqu'à $\lambda = 0,1$ côté Alex. Il semble donc qu'il y a une différence notable de spécificité de vocabulaire entre les corpus. Cette observation est soutenue par la différence de taille de leur vocabulaire : celui des données sur Alex compte 36755 entrées (pour 20016 tweets) et celui sur l'Ukraine 43367 (pour 18311 tweets). Des expérimentations sur d'autres jeux de données sont nécessaires pour savoir d'où provient cet écart, qui peut avoir des origines multiples comme son sujet, la durée ou encore la population concernée par l'évènement.

Pour $\lambda = 0,01$, on ne tombe jamais dans le cas où un tweet est assez proche d'un centroïde de cluster pour y être rattaché dans les données sur la tempête Alex. Dans les données sur l'Ukraine, cela ne se produit que dans un cas très particulier. Le corpus contient 263 messages identiques à un élément près : le nom du membre du congrès américain interpellé avec une demande d'envoi d'aide matérielle et humanitaire à l'Ukraine. Certains clusters ne contiennent que des exemplaires de ces messages et d'autres peuvent leur être attribuer même avec une valeur de λ basse.

Ainsi, si λ est trop bas, la structure finale du clustering peut donc être entièrement due à la procédure de création de nouveau cluster et à celle de ré-entraînement du modèle de plongement. En analysant plus finement les clusters, quelle que soit la valeur de λ, nous avons relevé que les documents les plus proches du centroïde de leur cluster ne sont jamais moins distants qu'une valeur limite. En excluant les clusters de taille 1, l'élément de cette "couronne" le plus proche du centroïde est à une distance de $\approx 0,016$ pour les données du corpus "Alex", même dans les clusters dont la taille se compte en dizaines. Dans le corpus "Ukraine", cette valeur est de $\approx 0,013$ en dehors des clusters ne contenant que des messages dupliqués. Si on les considère, elle descend à 0,003 dans des cas où le même membre du congrès est adressés plusieurs fois avec ces messages et nulle part ailleurs dans le corpus.

C'est une observation intéressante, pour laquelle nous ne parvenons pas à trouver d'explication facilement vérifiable. Notre intuition serait que la durée du modèle d'apprentissage définit une proximité limite en fonction de la taille de sa fenêtre contextuelle. En effet, le modèle fait toujours une modification lors d'une itération d'entraînement du fait des échantillons négatifs qu'il tire.

Ainsi, même d'hypothétiques documents ne contenant que des mots qui n'apparaissant qu'entre eux ne pourraient jamais être plus proches qu'une certaine distance. Celle-ci serait définie en fonction du nombre de passage sur une fenêtre du modèle, défini par son nombre d'époque et par la taille de la fenêtre, donnant le nombre de mises à jour pour un document. C'est presque le cas des 263 messages dupliqués dans le corpus Ukraine, qui contiennent toutefois certains éléments utilisés ailleurs dans le corpus comme `#putin`. La distance qui les sépare, même pour les messages étant des parfaites copies, est très faible mais pas nulle.

Des expérimentations rigoureuses, comparant différents paramétrages du système et le diamètre de ces "couronnes", pourraient apporter des éléments soutenant cette hypothèse

ou permettant de l'exclure.

Toutefois, la présence de cette "couronne" autour des centroïdes nous indique qu'il semble exister une limite en dessous de laquelle le seuil λ ne permet pas de raccrocher un message à un cluster. Il n'est pas possible en l'état d'affirmer avec certitude comment cette valeur peut être identifiée. Des explorations plus poussées sont nécessaires pour cela. Pour évaluer les autres composants de la méthode, il semblerait toutefois pertinent d'utiliser $\lambda > 0,02$ pour les deux jeux de données.

7.3.4 Ré-entraînement et re-partitionnement

Pour ré-entraîner le modèle, nous avons positionné le seuil temporel à une journée, et celui du nombre de différences notables en DF-IDF, *retrain*, à 15% de la taille du batch traité. La taille ici est celle en nombre de tweets. Lorsqu'un tweet comporte au moins un mot w tel que $DF - IDF(w, B, D)$ est significativement différent de $DF - IDF(w, B, D \setminus B)$, on augmente le décompte de ces différences pour B de 1. Pour ces premières expérimentations, nous considérons que l'écart est significatif à partir de 30%.

Au cours de nos expérimentations, le modèle a été ré-entraîné 6 fois pour le corpus sur la tempête Alex et 4 fois pour celui sur la guerre en Ukraine. C'est toujours le top temporel qui en est à l'origine, jamais le suivi en DF-IDF. En temps cumulé, ces procédures ont pris environ 40 secondes sur les données sur la tempête et environ 30 sur celles sur la guerre.

Il n'est pas possible de proposer d'analyse robuste de cette observation en l'état. Elle dépend en effet de trop de facteurs différents. Cela pourrait d'abord simplement être un cas particulier pour nos jeux de données. Leur vocabulaire pourrait simplement être suffisamment stable au fil du temps pour que le DF-IDF ne puisse pas permettre d'en mesurer le changement. Les valeurs de seuils pourraient également ne pas être adaptées à ces données. Il semblerait en effet naturel que les seuils *retrain* et celui qui définit un écart de valeur comme significatif soient spécifiques à chaque jeu de données.

Une autre explication pourrait être qu'on ne peut simplement pas mesurer la variation de la langue avec le DF-IDF sans une référence linguistique plus générale sur la langue traitée. Il faudrait dans ce cas changer la façon de mesurer l'évolution de cette dernière pour ne pas se reposer sur une ressource extérieure.

Une fois le modèle ré-entraîné, la partition doit être reconstruite dans le nouvel espace. Dans ce prototype, nous utilisons K-Means, K étant égal au dernier nombre de clusters connus. Indépendamment de la qualité des résultats obtenus, une observation à cette étape du processus s'impose d'elle-même. Pour les valeurs de λ basses, on obtient un grand nombre de clusters de taille 1, comme nous l'avons vu dans les tableaux de la figure 7.8. Cela n'est pas compatible avec le fonctionnement de K-Means, et allonge considérablement la durée de traitement. Dans les expérimentations, les re-partitionnements pour les données sur la tempête peuvent, au cumulé sur leurs 6 exécutions, durer jusqu'à 1h15 pour $\lambda = 0,01$, K-Means étant lancé avec $K = 8169$ pour 18800 tweets lors du dernier appel. Pour le corpus sur la guerre et le même seuil, les 4 K-Means supplémentaires durent environ 50 minutes, avec $K = 7513$ pour 16667 tweets pour le dernier appel.

Il semblerait donc que K-Means ne soit pas approprié pour la procédure de re-partitionnement. Il pourrait alors convenir d'employer une procédure par densité à la place. En effet, il serait possible d'utiliser les informations recueillies sur la partition pendant la procédure incrémentale pour utiliser une méthodologie basée sur la densité.

7.3.5 Défragmentation

Il n'est pas surprenant de constater que la défragmentation n'est pas très impactante sur les résultats finaux pour les basses valeurs de λ. Elle est réciproquement trop forte quand ce seuil est plus élevé, amenant à presque fusionner tout le corpus en un seul cluster pour les plus hautes valeurs de λ que nous avons essayées. Nous pouvons le constater dans les tableaux de la figure 7.8.

Pour $\lambda < 0,08$, la défragmentation est exécutée moins de 15 fois pour les deux corpus. Ce chiffre tombe même à 0 pour $\lambda \leq 0,02$, en dehors du cas exceptionnel des messages dupliqués dans le corpus sur la guerre Russo-Ukrainienne. À l'inverse, elle est appelée plusieurs milliers de fois pour $\lambda \geq 0,1$ pour le corpus sur la tempête, et pour $\lambda \geq 0,3$ pour celui sur la guerre. C'est à chaque fois le plus grand cluster qui est concerné par plus de 80% des appels à la défragmentation. Ce cluster pourrait constituer une sorte de "tout-venant" lorsque λ dépasse une certaine valeur.

Cela étant, pour que la défragmentation soit lancée, il suffit d'un tweet en commun entre deux clusters. Ainsi, qu'il existe un cluster tout-venant ou non, la faiblesse de cette condition pose problème. En effet, il apparaît naturel qu'une valeur de λ trop élevée fasse s'agglomérer toute la partition, d'où les résultats pour $\lambda \geq 0,3$. Toutefois, il est impossible de savoir avec certitude si la taille des grands clusters obtenus pour $0,08 \leq \lambda \leq 0,2$ est due au seuil étant trop permissif ou à la fusion étant exécutée un trop grand nombre de fois. Cette incertitude est d'autant plus problématique en constatant, dans le tableau 7.8b, la présence d'autres clusters dont la taille se compte en milliers pour une grande valeur de λ.

7.3.6 Descriptions

En raison du grand nombre de clusters par valeur de λ, il est délicat de présenter les descriptions qui sont construites. Nous présentons donc les observations les plus notables sur le contenu des descriptions.

Les descriptions sont construites de la façon décrite dans la section 7.2.6, avec $\gamma = 10$. Les mots des descriptions sont donc issus des 10 documents les plus proches du centroïde de leur cluster.

Les tableaux 7.1 et 7.2 montrent un échantillon des descriptions obtenues avec notre prototype de description incrémentale. Nous présentons toujours les clusters de plusieurs milliers de tweets pour une valeur de λ, puisqu'ils sont peu nombreux. Les clusters présentés dont la taille se chiffre en centaines ou en dizaines ont été choisis aléatoirement parmi ceux dont la taille est du même ordre de grandeur. Les valeurs du seuil λ présentées ont été sélectionnées car le nombre et la taille des clusters obtenus est proche de ceux obtenus avec une valeur de λ similaire.

Même si elle est aléatoire, cette sélection nous permet de présenter toutes les tendances que nous avons pu observer dans les résultats.

Nous pouvons remarquer quelques tendances générales. D'abord, certains éléments de descriptions se retrouvent presque systématiquement dans les grands clusters. Pour les données sur la tempête Alex, c'est le cas par exemple d'éléments de la comptine que nous avons évoqué en Section 5.3.1.1 comme `dansaient capucine vivement`, `tout` et `amande bois`. Dans les données sur la guerre, on peut considérer les 263 messages dupliqués comme l'équivalent de cette comptine en terme de redondance et d'impact sur l'espace de repré-

Valeur de λ	taille du cluster	description
0,02	4146	france sinistre, mesure exceptionnel, trois, #peinture, dansaient capucine vivement, tout, amande bois, soucoupe
0,02	131	apre destruction, vigilance rouge levee, alpes-maritime, reconforter sinistre, sept, parc animalier, voici attendre vendredi, portee disparue, plusieurs personne, helicoptere, visite
0,02	37	mobilise enfant, alpes-maritime, terrible image degat, jacob, devastee video, tout ecole, envoye, detresse, apre crue, retablie
0,05	8431	trois, bout dune brancher, dansaient capucine vivement, ciel bleu, artificiel, proteger, premier fois, #peinture, image aerien, soucoupe, tente
0,05	10	rmcsport, charlotte, tres beau, boxer, coincee, lycee, suite j'attend, trop, genre, concentrer, fenetre, dire vouloir, conduire sous, prendre
0,05	26	bretagne, vigilance rouge morbihan, mort apre passage, foyer sans courant, france priver, cinq departement, vigilance orange vent violent, cest horrible, foyer, retrouve courant
0,1	14101	ciel bleu, marron bois, tout, bout dune, pousse dansaient, artificiel, proteger, premier fois, daxa, france sinistre, image aerien, #peinture
0,1	14	snaper, lexterieur, aller venir bouffer, veterinaire, apre destruction, parc loup, utiliser, propre, tweet, contre, littoral, helicoptere, rescape, eclatee, tout, lhistoire, jeudi soir, plus, pluvio, encore debout, vrai, plus
0,1	19	georgioguillon, briquetroly, matchedNumber, patrickbruel, merci aussi, matchedNumber, semer vent recolter, calme semi, chaleur, commencer, sombrer, routine, oublier, jean castex, ministre, grand, oublie, tout celui
0,3	19574	doseille, gros cheque, perdu tout, reserve, foin clicanimaux, #peinture, #painting, avocat nice, service sinistre, dautomne, region quebec, #meteoqc
0,3	15	j'arrive, deter, connu, pire, finistere, comme, gosse, lannonce, bleue80243434, centaine, soldat, tues, #bretagne pays, loire, defense137, tout, unite, s'evacue, d'mon, corps creer, songe, bizarrement, rien, propose, comme, grave degat, l'arriere pays nicois,

TABLEAU 7.1 – Échantillons des descriptions obtenues avec la description incrémentale dans les données sur la tempête Alex, avec la valeur de seuil de lambda associé et la taille du cluster décrit

sentations des mots. En revanche, c'est une facette de l'évènement que l'on chercherait à qualifier pour le corpus "Ukraine". Le lobbying pendant un évènement géopolitique majeur peut en effet en être considéré comme une facette ou un sous-évènement important à repérer. La comptine s'apparente plutôt à du bruit dans nos données, au regard de la catastrophe naturelle en cours.

Cette observation pourrait s'expliquer du fait de l'extrême spécificité des composants de ces messages. Les membres de ces messages peuvent former des UMMs de deux voire trois mots, pour ceux qui n'apparaissent pas ou peu dans d'autres messages. Une telle spécificité pourrait amener les mots de ces documents à rester relativement proches du point neutre de l'espace. En effet, comme ils n'apparaissent avec aucun autre mot pouvant les rapprocher d'une certaine zone sémantique ou d'une autre, ils pourraient ne pas s'éloigner de l'origine. Un cluster de grande taille aurait ainsi de bonnes chances de les englober. Plus ce cluster est grand, plus il couvre l'espace de représentation. Comme les distances sont relativement faibles, le centroïde de ce grand cluster peut ainsi être assez proche du point neutre, et les documents les plus proches sont alors ceux de ces cas particuliers.

Nous observons une autre tendance générale. Les grands clusters ont une inclination à avoir des descriptions plus courtes. Les clusters dont la taille se compte en dizaines ont en moyenne entre 20 et 30% de descripteurs supplémentaires que ceux dont la taille se chiffre en milliers (et parfois même plus dans quelques cas). Une raison identifiable pourrait être que les 10 documents les plus proches du centroïde du cluster sont beaucoup plus variés pour les petits que les grands clusters. Comme les descripteurs doublons sont retirés, les descriptions sont ainsi mécaniquement plus longues pour les petits clusters.

Nous ne trouvons pas d'explication plus robuste à cette observation.

Une observation très clairement visible est qu'une valeur de λ trop basse éparpille l'information entre les clusters. Dans le tableau 7.1, on peut par exemple voir `alpes-maritime` dans les clusters de taille 131 et 37 pour $\lambda = 0,02$. De la même manière, on peut voir le tag `humanitarian assistance defend` dans les deux clusters du tableau 7.2 pour $\lambda = 0,02$. Il est délicat de quantifier le nombre de cas où cette séparation à lieu, mais son observation est cohérente avec le très grand nombre de clusters de petite taille observé quand la valeur de λ est basse. Elle est également cohérente avec la mécanique de construction des vecteurs de documents. Si le seuil λ est très bas, alors un variation même légère de leur vocabulaire peut amener deux documents à être dans des clusters séparés.

Nous partageons une dernière observation. Les UMMs sont assez fréquentes parmi les descripteurs qui sont utilisés. À l'échelle des partitions, elles représentent environ 10% des descripteurs pour les deux corpus. Si les tableaux 7.1 et 7.2 peuvent donner l'impression qu'elles sont plus utilisées que cela, c'est parce que les UMMs sont plus utilisées dans les descriptions des grands clusters que dans celles des petits. Selon notre intuition, les grands clusters semblent regrouper les tweets ayant un vocabulaire à la limite d'être considéré comme trop spécifique pour avoir un intérêt descriptif. Cette hypothèse pourrait être confirmée avec des expérimentations mesurant la proportion de vocabulaire dont la valeur de I' est proche de cette limite.

Nous pouvons remarquer que, indépendamment de leur fréquence, les UMMs montrent une nouvelle fois leur intérêt descriptif. Elles sont en effet pour la plupart faciles à interpréter, font référence à une entité nommée ou sont des collocations faisant sens. Dans les données sur la tempête Alex, on retrouve par exemple `france sinistre`, `mesure exceptionnel`, `portee disparue`, `reconforter sinistre`, `terrible image degat`, `semer vent recolter` ou encore `vigilance orange vent violent`. Dans les descripteurs des données sur l'Ukraine, nous pouvons voir par exemple `stop russian aggression`, `solidarity people`, `sanction russian oligarch`, `going long`, `neighbouring country` ou encore `declare state emergency`.

On les retrouve également dans les grand clusters, avec des descripteurs comme `humanitarian assistance defend`, `stop innocent civilian`, `death` et `potus provide`.

Valeur de λ	taille du cluster	description
0,02	114	ukrainian russia, within hour, create russia stop, collectible, humanitarian assistance defend, send love, nottingham, johnboozman, solidarity people
0,02	651	need, weapon, humanitarian assistance defend, #putin, thisiskyler, protect property, stop innocent civilian, death, potus provide
0,05	2418	need, weapon, humanitarian assistance defend, #putin, stop innocent civilian, death, potus provide, #stopputin
0,05	446	#kharkiv #kyiv, #odesa, #trostyanet, #sumy, #severinovka, stop russian aggression, nato, close exclude, russia security council, winner, ukariene, supprt army, csparks1234, jamertonia, manolo_realengo, nope, read history, parliament vote, declare state emergency, russia recognize, separatist region, help, protect ukrainian city, russian, missile provide
0,05	16	medvedevrussiae, mean, #russian, started, #fuckputin, instituion, produce, engineer, real life, going lose, approve, state emergency, start midnight, check, public, transport, limit, four, unarmed, black, kill, first, dominate, social medium
0,3	8137	need, weapon, humanitarian assistance defend, member resistance, sanction russian oligarch, georgia moldova, apply join, make donation help, chrisvanhollen, denizensbrewe, conflict', impact, crypto market
0,3	5033	leave poland, monday night, #kharkiv #kyiv, #odesa, zelenskyyua, umlandandrea, send weapon, lend, lease, please stay safe, need ever, whataboutary, merely, call hypocrisy, justify invasion, organise, evacuation, neighbouring country
0,3	124	russia, coulda, superstar, space, #wwiii, fake misinformation, neutrality, provocation, president jewish, people, around world, russia, already lose, wounded, bomb hospital
0,3	19	montreal, winnipeg, stand foot, like, senseless, african, going long, pray, morning, suisse, scamme, investor, receive, roku, pypl, #facebook, #bitcoin, #swiss, #technology
0,5	13947	need, weapon, humanitarian assistance defend, #putin, stop innocent civilian, conductor, driver, throw, jail, zelenskyyua potus
0,5	2473	putin, macron, worst come, russia condition, together, courage, respond move, meeting, distribute, information russian, girl kill, strike, choice learn, captured, putin, biden russia, forget mention, banking system, fail, pretend
0,5	220	hope could, want, review, world matchedNumber, want, thermonuclear, pharma, sector, putin russia, euromaidanpress, news russia, situation tune, advance, shake hand, hero, russia', perfect, condition, outbreak, people
0,8	17109	need, weapon, humanitarian assistance defend, #putin, stop innocent civilian, death, potus provide, #stopputin, train, conductor, driver, throw, jail, zelenskyyua potus
0,8	515	ursula, help putin, finance, want, invade, question, crisis, putin, need stopped, czarist, russia border, connection, 'holy, russia', ussr, savage, unnecessarily, admit, hospital bomb, russian agent, trial crime, sovereign country, join whatever,
0,8	23	know, little, thank, russian, people, twitter right, protest, show, close airspace, america, nato, center, till, american, know, heath, supportd, undercover, asassination, catholic, duffy, form, fascist, #violence, #victim, #military, #rebellion, first ever, collector, dear friend, give chance, open, hand, would, stop, school, help ukrainian

TABLEAU 7.2 – Échantillons des descriptions obtenues avec la description incrémentale dans les données sur la guerre Russo-Ukrainienne, avec la valeur de seuil de lambda associé et la taille du cluster décrit

7.4 Limitations et ouvertures

Le prototype implantant l'architecture que nous avons présentée en section 7.2 permet d'obtenir de premiers résultats. Ces derniers, s'ils doivent être pris en compte en gardant à l'esprit qu'ils restent exploratoires, permettent d'identifier des limitations dans cette architecture et les principes qu'elle met en oeuvre.

Les deux principaux problèmes mis en évidences de manière robuste sont dus au re-partitionnement et à la défragmentation.

Re-partitionnement Utiliser K-Means pour recalculer la partition quand le modèle de plongement est ré-entraîné pose un problème pragmatique et un autre plus conceptuel. La description incrémentale peut détecter un grand nombre de clusters, en fonction de la valeur du seuil λ. Toutefois si K-Means et exécuté avec une grande valeur de K, cela allongera considérablement la procédure de mise à jour. Cela peut résulter, comme nous l'avons vu, en un temps de réponse par rapport au flux trop important.

Le problème conceptuel réside dans le fait qu'utiliser le dernier nombre de clusters connu n'est pas forcément adéquat dans le nouvel espace de représentation. Le nombre "réel" d'objets sémantiques dans le corpus ne change jamais en fonction de l'état du modèle de plongement. Il ne dépend que des données brutes. Mais le nombre qu'il est possible d'en identifier à travers le découpage de l'espace de représentation est lui basé sur l'état de Doc2Vec. Plus précisément, il est basé sur la similarité que l'on peut mesurer dans cet espace et sur le positionnement des vecteurs de documents.

Ainsi, un groupement de documents précédemment identifié peut ne plus l'être avec l'incorporation de nouvelles données. Ces dernières ont deux type d'impact sur la partition. D'abord, les associations mots-contextes apprises dans celles-ci changent l'impact des mots, même ceux déjà connus, sur leurs documents. Les nouveaux éléments de vocabulaire ont également cet effet. Ensuite, le positionnement des nouvelles instances dans la partition change mécaniquement la structure de cette partition.

Ce problème de re-partitionnement peut être abordé de deux façons.

D'une part, on peut envisager de considérer les tweets isolés comme du bruit. Cela ramènerait la valeur de K à un ordre de grandeur pour lequel le temps de calcul de la partition est acceptable. Cette approche s'éloignerait du parti pris initial de nos travaux, où toutes les données d'un corpus sont considérées de manière égalitaire. Il est toutefois reconnu que les données issues des réseaux sociaux sont notablement bruitées. Elle pourrait donc être acceptable, à condition de mener une étude approfondie de l'effet qu'aurait la valeur de λ sur l'identification du bruit dans les données.

On répond ainsi au problème de temps de calcul, mais pas à celui de la détection de l'éventuelle nouvelle structure dans les données.

D'autre part, un autre algorithme de clustering pourrait être employé. La principale raison ayant motivé l'utilisation de K-Means au fil de nos travaux est le fait qu'il ne nécessite pas de connaissance préalable sur les données à partitionner en dehors de K. Toutefois, au moment où il faut partitionner le nouvel espace de représentation, on dispose d'informations au sujet de cet espace. Elles sont issues de la partition de l'ancien espace de plongement. Comme l'espace a changé, il faut considérer ces informations comme des aperçus. Elles sont toutefois les plus précises et les plus proches des données dont on peut disposer au moment de re-partitionner. Une série d'expérimentations évaluant la façon

dont ces informations évoluent au fil de la collecte des données permettrait de vérifier si elles sont utilisables à cette fin.

En particulier, il serait possible de s'appuyer sur ces connaissances pour découvrir le nombre de clusters dans le nouvel espace avec un algorithme travaillant à base de densité. On trouve par exemple parmi eux DBSCAN, présenté en section 2.1.5. Il nécessite deux paramètres. D'abord, le nombre minimum d'instances devant appartenir à un même voisinage pour former un cluster. Ensuite, la distance maximum pouvant séparer deux instances que l'on considère appartenir au même voisinage.

L'utilisation d'un tel algorithme tend à être soutenue par l'observation de la taille minimum de la couronne des clusters. Si d'autres expérimentations confirment l'existence de cette limite minimale de taille et mettent en évidence un moyen de l'identifier, il serait possible de s'en servir pour établir les paramètres d'un algorithme comme DBSCAN. Cela amènera toutefois également de nouvelles problématiques, du fait que ce type de clustering ne construit pas des clusters sphériques mais dont les formes suivent les zones de forte densité.

Défragmentation La défragmentation mise en oeuvre ici pose problème. En particulier, elle limite les analyses que l'on peut faire des résultats pour les valeurs relativement élevées de λ. Cela est dû à sa condition de déclenchement, beaucoup trop faible en l'état. Une première façon d'adresser ce problème serait de devoir compter au moins un certain nombre de tweets pouvant appartenir à plusieurs clusters, fonction de la taille de ces derniers, pour lancer la défragmentation.

Une autre façon de décider de la lancer serait d'établir une mesure de similarité entre clusters. Des clusters suffisamment proches pourraient ainsi être identifiés et fusionnés. Pour mettre en oeuvre un fonctionnement de ce type, il faut établir une façon de représenter le cluster pour pouvoir le comparer aux autres. Nous utilisons les centroïdes, sous la forme des points moyens, des clusters pour les comparer aux nouveaux messages. Comparer deux centroïdes semble toutefois limité si l'objectif est de décider de leur fusion. Ce type d'approche devrait s'appuyer sur un seuil de distance séparant les points des clusters, en dessous duquel on déciderait leur fusion. Il conviendrait donc d'établir une comparaison de cluster plus raffinée.

Gestion de la mémoire Un point que nous n'avons pas commencé à traiter ici est la gestion de la mémoire. Comme la description incrémentale écoute des données en continu, elle peut être amenée à agréger une quantité de données importante. Si ce problème est issu d'une problématique purement matérielle, un autre plus complexe se pose pour la même raison. Un sous-évènement représenté par un cluster auquel on n'attribue plus de tweets depuis une durée importante doit-il toujours être décrit ? Réciproquement, on peut capter un message qui aurait être agrégé à un cluster qui a éventuellement été retiré de la partition car il n'a pas été alimenté pendant trop longtemps.

La question de l'obsolescence des clusters et de leur description se pose du fait que les données traitées sont issues d'un flux, et sont donc relatives à leur date de parution. Cette problématique devra être abordée par la solution qui sera proposée pour procéder à la description incrémentale.

Description Les observations que nous avons pu faire des descriptions obtenues avec ce prototype incrémental sont délicates à analyser. Certaines tendances générales se remarquent, comme nous avons pu le voir dans la section 7.3.6. Les UMMs facilitent l'interprétation des descriptions mais il y a des cas où il reste difficile d'attribuer un sens à une description.

Toutefois, les problèmes liés au clustering empêchent de savoir si ces difficultés d'interprétations viennent de la méthodologie de construction des descriptions ou de la partition elle-même. Il convient donc de traiter les problèmes de partitionnement d'abord, pour pouvoir établir des analyses des descriptions sur une base robuste, et d'adresser leurs éventuels problèmes ensuite.

7.5 Conclusion

Ce chapitre présente les travaux exploratoires que nous avons menés pour transposer l'approche que nous avons présentée dans les chapitres 5 et 6 dans un contexte incrémental. Nous proposons un ensemble de procédures pour s'adapter au fait que les données soient reçues en continu et une première architecture pour les éprouver. Nous obtenons des résultats qui montrent la faisabilité de ce portage.

Le modèle Doc2Vec reste approprié au contexte dynamique. Les vecteurs qu'il apprend sur les corpus initiaux de nos expérimentations permettent de construire des partitions de manière stable. Il apparaît donc que Doc2Vec puisse s'entraîner de manière convenable sur un corpus encore plus restreint. De plus, sa légèreté est compatible avec les besoins de ré-entraînements réguliers. Chaque entraînement de zéro du modèle n'a en effet jamais pris plus de 8 secondes.

De plus, la modification légère de l'implantation de Doc2Vec que nous avons effectuée permet de mettre en oeuvre le même raisonnement que celui s'appuyant sur l'ablation de mots, présenté dans le chapitre 6. Elle présente l'avantage significatif de pas nécessiter plusieurs inférences pour un même vecteur de document, permettant d'identifier des mots sans intérêt descriptif pour un cluster sans sur-coût de calcul.

Le clustering permet toujours d'identifier des messages dont la constitution est similaire. La représentation en espace sémantique semble donc permettre de comparer vecteurs de documents et représentants de clusters aussi bien dans un contexte statique que dynamique. La façon dont le clustering est utilisé dans ce prototype semble toutefois limitée. Le calcul de la partition dans l'espace de représentation issu de la mise à jour du modèle de plongement au cours de l'écoute pointe plusieurs problèmes.

L'utilisation d'un seuil λ de proximité pour décider d'attribuer une nouvelle instance à un cluster fait apparaître de nombreux clusters de petite taille, quand la valeur de λ est basse. Cette constatation pourrait indiquer qu'il nous faut reconsidérer la façon d'envisager les données, en particulier en redéfinissant ce que nous considérons comme du bruit.

Avant de pousser cette réflexion, il faut dans tous les cas mener des efforts pour que la construction de la partition soit plus robuste. En particulier, les procédures de défragmentation et de création de nouveaux clusters nous semble les plus prioritaires à améliorer. Le principal problème de notre défragmentation actuelle est que son seuil de déclenchement est trop permissif. Ce point est d'autant plus problématique lorsque l'on traite de données textuelles où l'utilisation du vocabulaire est aussi diffuse, comme c'est le

cas pour les données issues des réseaux sociaux. La première étape est ainsi naturellement de durcir cette condition. Une série d'expérimentations adaptées permettra de mettre en évidence si le problème se limite à une valeur de seuil, ou si c'est le processus même de fusion de cluster pour la défragmentation qui n'est pas adapté.

La création de cluster est également à revoir. Créer un nouveau cluster pour intégrer une nouvelle instance trop éloignée de ce que l'on connaît déjà semble une façon appropriée de procéder. En revanche, raccrocher toutes les instances existantes à un tel nouveau cluster si elles se trouvent plus proches de lui que du centroïde de leur cluster actuel semble également trop permissif. Il faut chercher une nouvelle procédure pour opérer cette étape.

Conclusion

À travers cette thèse, nous avons introduit une approche pour explorer et décrire des données issues des réseaux de micro-blogging. Nous avons présenté les difficultés qui se posent lorsque l'on traite de contenu rédigé en langue naturelle et ce que peut apporter la recherche de structure dans ce type de données. Nous avons utilisé un modèle d'apprentissage pour calculer des représentations de textes courts constituant nos corpus d'étude, qui permettent de les traiter dans le cadre d'une sémantique distributionnelle. Il est alors possible d'utiliser des méthodologies de fouille de données pour détecter les groupes de messages de sens similaire pour décrire un corpus aspect par aspect. Les descriptions sont alors construites à partir des mots qui sont pertinents et représentatifs par rapport à l'ensemble de micro-posts décrit, le modèle de plongement que nous avons utilisé permettant de détecter des descripteurs plus longs et plus spécifiques à l'ensemble de messages décrits. Enfin, nous avons exploré plus finement l'espace de représentation appris par le modèle de plongement pour détecter des descripteurs plus longs et plus spécifiques à l'ensemble de messages décrits.

Traiter automatiquement des données textuelles rédigées en langue naturelle est une tâche complexe. La principale difficulté de nos travaux vient du fait que les corpus que nous traitons sont relativement restreints en taille et que la langue dans laquelle ils sont construits évolue constamment du fait de son utilisation en ligne. De plus, puisque l'application de ces travaux est prévue à travers leur déploiement sur des machines de puissance modeste, la solution proposée doit donc également avoir un coût computationnel maîtrisé.

Ainsi, la première étape de nos travaux a été de valider nos hypothèses de travail avec le modèle correspondant à nos contraintes. Cette contribution, consistant en un système bout-en-bout et ses résultats expérimentaux, a fait l'objet des publications à CNIA 2022 et ICTAI 2022. Cette dernière présente l'approche que nous avons construite, basée sur le modèle de plongement Doc2Vec, qui est peu profond et léger à entraîner. Cette approche, évaluée sur des jeux de données figés, nous a permis de montrer que Doc2Vec permet de construire des vecteurs de mots et documents robustes. Ces derniers peuvent être explorés via une méthodologie de clustering pour identifier des structures dans les corpus, consistant ici en des groupements de tweets au sens similaire. Une fois ces clusters identifiés, le système y identifie des mots adaptés à leur description qui deviennent des candidats descripteurs. Ceux-ci sont alors transmis à un modèle déclaratif d'assignation, qui construit les descriptions des clusters en fonction d'un ensemble de contraintes de qualité. Comme les métriques d'évaluations portées à notre connaissance ne peuvent pas s'appliquer ou perdre de leur valeur dans le cas de nos travaux, nous en avons également proposé deux et les avons appliquées sur nos résultats. Ces derniers montrent que notre système est adapté à la description de micro-posts, et permet de rendre leur contenu accessible à un utilisateur humain.

La principale limitation de cette approche réside dans le manque de spécificité des descripteurs, du fait qu'ils se limitent à des mots privés de leur contexte. Ces travaux sont menés avec l'objectif de ne pas faire appel à des ressources ou à des connaissances extérieures aux données. Cela permet d'en limiter la complexité de mise en oeuvre et de la garder la plus agnostique possible par rapport aux données traitées. Nous avons donc cherché à exploiter le modèle de plongement faisant déjà partie de notre architecture, pour identifier quelles parties d'un voisinage de mots au sein d'un message devraient être considérées ensemble. Cette contribution à la détection de descripteurs plus spécifiques pour un groupement de micro-posts a fait l'objet d'une publication à ICTAI 2023. Elle consiste à corréler les pondérations des contextes des mots apprises par Doc2Vec aux représentations même des mots pour identifier les associations mots-contextes de poids élevé. Les résultats que nous avons obtenus montrent que cet "essorage" du modèle de plongement présente un intérêt descriptif certain en associant les mots qui forment des unités au sens cohérents. Nous parlons d'un "essorage" car cette approche consiste intuitivement à presser les couches du modèles entre elles pour identifier les zones fortement pondérées. Les zones faiblement pondérées pour un mot donné ne sont pas considérées, et donc évacuées. Ce processus, corrélé à la topographie de la partition des documents, permet d'identifier les mots ne présentant pas d'intérêt descriptif pour un cluster.

Finalement, la dernière étape de cette thèse s'est matérialisée par de premiers travaux exploratoires visant à passer à la gestion des flux pour faire de la description incrémentale de données. Si l'architecture que nous avons mise en place nous a permis d'éprouver nos hypothèses au contact de données réelles et de montrer l'intérêt de notre approche, certains de ses composants ne peuvent pas fonctionner avec des corpus dynamiques. Cette transposition soulève en outre de nouvelles problématiques et de nouvelles difficultés qu'il faut aborder en complétant le système. Nous avons donc proposé une procédure, munie d'un prototype que nous avons testé en simulant un flux de données avec nos corpus d'étude précédents. Les résultats obtenus sont exploratoires et d'autres seraient nécessaires pour valider la robustesse des analyses qu'il permettent de faire. Ils sont toutefois encourageants car ils indiquent que le portage de notre approche sur des données en flux est possible. Les descriptions obtenues en l'état manque en clarté mais rendent bien compte des objets sémantiques intéressants dans nos corpus, et elles sont construites de manières incrémentale.

Ces résultats mettent également en évidence les limitations les plus visibles de l'architecture. La défragmentation de la partition est un enjeu critique de toute procédure incrémentale. Dans notre prototype, la procédure répondant à ce besoin est déclenchée par une condition trop permissive. Elle fusionne de ce fait des éléments qui pourraient rester séparés. En outre, la procédure de re-construction de la partition suivant le ré-entraînement du modèle de plongement n'est pas adaptée au cadre incrémental. En effet, mettre à jour Doc2Vec pour encoder de nouveaux éléments dans son espace implique de le ré-entraîner. Dans le prototype, K-Means est employé cette étape pour re-partionner le nouvel espace de représentation. Il apparaît que soit cet algorithme n'est adapté pour cette procédure, soit il faut changer la façon de considérer les messages isolés dans cet espace. Ces derniers font en effet exploser le nombre de cluster à rechercher dans les cas où le seuil d'attribution d'un message à un cluster est bas.

Ces résultats ouvrent ainsi des pistes pour poursuivre nos travaux.

Perspectives

La perspective à plus court terme pour nos travaux est de continuer le travail d'adaptation au contexte dynamique.

Exploration par densité Le principal problème de notre architecture incrémentale dans son état actuel est la composition de la partition au fil de la réception des messages. Celle-ci s'appuie en particulier sur un seuil de proximité, qui permet de décider si un nouveau message est assez proche d'un groupement déjà connu ou s'il faut en créer un nouveau pour pouvoir l'y intégrer. Ce type d'approche peut amener à créer des clusters séparés alors qu'ils devraient n'en former qu'un à cause de l'ordre dans lequel les données sont reçues. Le composant de notre prototype n'offrant pas une réponse satisfaisante à ce problème, nous prévoyons d'en changer le fonctionnement pour tester ce que permettrait d'obtenir une exploration basée sur la densité dans l'espace exploré. Au cours de nos travaux, nous n'avons jamais pu recourir à de telles méthodologies puisque nous ne disposions pas d'information a priori sur cet espace. Le contexte incrémental change cet état de fait. Le traitement de flux requiert de partir d'une base et d'explorer les données au fil de leur réception. Ainsi, au moment de devoir décider la fusion de certains clusters, on dispose de la connaissance de l'espace de plongement acquise au cours du traitement incrémental. On peut donc utiliser une exploration basée sur la densité.

Les résultats obtenus en mettant à profit ce type de méthodologie produiront de nouveaux résultats. Leur évaluation pourrait permettre d'identifier si les limitations de notre architecture incrémentale sont dues à un défaut de partitionnement ou de construction de description.

Évaluation sur de nouveaux jeux de données Il faudra éprouver sur de nouveaux jeux de données notre système implantant une architecture de description incrémentale. Les expérimentations au cours de cette thèse avaient été simplifiées par la mise à disposition du monde académique d'une grande quantité de données Twitter gratuites. Le changement de politique récent de la plate-forme limite à nouveaux son utilisation. Des travaux récents, comme ceux de (Meunier et al., 2023), ont toutefois rendu disponible de nouvelles ressources. Celles-ci permettent de vérifier la validité et la robustesse de nos travaux face à d'autres approches récemment parues, et de les évaluer avec de nouvelles métriques.

Il s'agit d'une opportunité très intéressante que nous souhaitons explorer.

Comparaison à d'autres ressources linguistiques Au cours de nos travaux, la dynamique d'évolution de langue pratiquée sur les réseaux sociaux a été une contrainte à laquelle il a fallu s'adapter. La démocratisation des modèles de plongements, en particulier des modèles profonds de très grande taille, a fait apparaître plusieurs questions sur la capture de la sémantique qu'ils opèrent. Des travaux comme ceux de (Ferret, 2022) montrent qu'il existe des interactions à explorer entre les plongements construits par ces grands modèles et ceux, statiques, construits par un modèle comme Doc2Vec.

Nous avons montré, avec notre exploitation de ce dernier, qu'il est possible de tirer parti des modèles de plongements différemment qu'à travers la seule utilisation de leurs

vecteurs. Nous souhaitons donc, à plus long terme, explorer finement la comparaison de la sémantique apprise par les grands et les petits modèles de plongement.

La sémantique des descriptions est également à explorer plus en détail dans de futurs travaux. Cet axe est à mener en parallèle d'une étude plus fine de la granularité d'évènements à identifier pour constituer les facettes de l'évènement cible. Cette étude aura de plus vocation à permettre d'identifier ce qui caractérise le bruit d'un élément à décrire au sein d'un corpus. Les interactions linguistiques entre les éléments d'une description constituent de bons candidats à explorer dans ce sens.

Pour cela, des ressources ontologiques permettant d'identifier les relations entre les éléments des descriptions peuvent être exploitées. Cela permettrait en effet une caractérisation plus raffinée de ces dernières. Les ontologies mobilisées, externes et donc relevant d'une connaissance non spécifique à un corpus, pourraient être comparées à des ontologies déduites des descriptions, spécifiques à un jeu de données.

Bibliographie

Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W., and Hassan, A. (2022). Topic modeling algorithms and applications : A survey. *Information Systems*, page 102131.

Ackerman, M. and Dasgupta, S. (2014). Incremental clustering : The case for extra clusters. *Advances in neural information processing systems*, 27.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, Chile.

Aguilar, G., Maharjan, S., López-Monroy, A. P., and Solorio, T. (2017). A Multi-task Approach for Named Entity Recognition in Social Media Data. In *W-NUT*.

Alguliev, R. M., Aliguliyev, R. M., Hajirahimova, M. S., and Mehdiyev, C. A. (2011). Mcmr : Maximum coverage and minimum redundant text summarization model. *Expert Systems with Applications*, 38(12) :14514–14522.

Arthur, D. and Vassilvitskii, S. (2007). K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035.

Badjatiya, P., Kurisinkel, L. J., Gupta, M., and Varma, V. (2018). Attention-based Neural Text Segmentation.

Baghel, R. and Dhir, R. (2010). A frequent concepts based document clustering algorithm. *International Journal of Computer Applications*, 4(5) :6–12.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*.

Balakrishnan, V. and Lloyd-Yemoh, E. (2014). Stemming and lemmatization : A comparison of retrieval performances.

Becker, H., Naaman, M., and Gravano, L. (2011). Beyond trending topics : Real-world event identification on twitter. In *Proceedings of the international AAAI conference on web and social media*, volume 5, pages 438–441.

Beil, F., Ester, M., and Xu, X. (2002). Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442.

Beliga, S., Meštrović, A., and Martinčić-Ipšić, S. (2015). An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*, 39(1) :1–20.

Bender, E. M. and Lascarides, A. (2020). Collocations and other multiword expressions. In *LFNLP*.

Blei, D. M. (2003). Latent Dirichlet Allocation. *JMLR*.

Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *arXiv :1607.04606 [cs]*. arXiv : 1607.04606.

Boudin, F. (2013). A comparison of centrality measures for graph-based keyphrase extraction. In *Proceedings of the sixth international joint conference on natural language processing*, pages 834–838.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7) :107–117.

Bruni, E., Tran, N.-K., and Baroni, M. (2014). Multimodal distributional semantics. *Journal of artificial intelligence research*, 49 :1–47.

Buckland, M. (2018). Document theory. *KO Knowledge Organization*, 45(5) :425–436.

Calzolari, N., Fillmore, C. J., Grishman, R., Ide, N., Lenci, A., MacLeod, C., and Zampolli, A. (2002). Towards Best Practice for Multiword Expressions in Computational Lexicons. *LREC*.

Can, F. (1993). Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems (TOIS)*, 11(2) :143–164.

Charolles, M. (1995). Cohésion, cohérence et pertinence du discours. *Travaux de Linguistique : Revue Internationale de Linguistique Française*, (29) :125–151.

Chauhan, U. and Shah, A. (2021). Topic modeling using latent dirichlet allocation : A survey. *ACM Computing Surveys (CSUR)*, 54(7) :1–35.

Cori, M. and Léon, J. (2002). La constitution du TAL.

Culmer, K. and Uhlmann, J. (2021). Examining lda2vec and tweet pooling for topic modeling on twitter data. *Wseas Trans. Inf. Sci. Appl.*, 18 :102–115.

Curiskis, S. A., Drake, B., Osborn, T. R., and Kennedy, P. J. (2020). An evaluation of document clustering and topic modelling in two online social networks : Twitter and Reddit. *IPM*.

Dao, T.-B.-H., Kuo, C.-T., Ravi, S. S., Vrain, C., and Davidson, I. (2018). Descriptive Clustering : ILP and CP Formulations with Applications. In *IJCAI*.

Davidson, I., Gourru, A., and Ravi, S. (2018). The Cluster Description Problem - Complexity Results, Formulations and Approximations.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2) :224–227.

Dawei, W., Alfred, R., Obit, J. H., and On, C. K. (2021). A literature review on text classification and sentiment analysis approaches. *Computational Science and Technology : 7th ICCST 2020, Pattaya, Thailand, 29–30 August, 2020*, pages 305–323.

de Saussure, F. (1971). *Cours de linguistique générale (1916)*. Payot.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv*.

Dowden, B. H. (1993). *Logical reasoning*. Bradley Dowden.

Elguendouze, S., de Souto, M. C., Hafiane, A., and Halftermeyer, A. (2022). Towards explainable deep learning for image captioning through representation space perturbation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fathi, M. and Bevrani, H. (2019). *Artificial Intelligence and Evolutionary Algorithms-Based Optimization*, pages 137–169.

Ferret, O. (2022). Décontextualiser des plongements contextuels pour construire des thésaurus distributionnels. In *29e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2022)*, pages 315–324.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context : The concept revisited.< italic> acm transactions on information systems</italic>, 20 (1) : 116-131. *Google Scholar Google Scholar Digital Library Digital Library*.

Firth, J. R., Haas, W., and Halliday, M. A. (1968). *Studies in linguistic analysis*. Blackwell.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2 :139–172.

Flores, S. (2019). Variational Autoencoders are Beautiful | Blogs.

Gao, Y., Zhao, W., and Eger, S. (2020). SUPERT : Towards New Frontiers in Unsupervised Evaluation Metrics for Multi-Document Summarization. *ACL*.

Gardiner, A. H. (2016). Définition du « mot » et de la « phrase ». *Modèles linguistiques*, XXXVIII(74) :97–108.

Gladkova, A. and Drozd, A. (2016). Intrinsic Evaluations of Word Embeddings : What Can We Do Better ? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42, Berlin, Germany. Association for Computational Linguistics.

Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*.

Gracianne, O., Lefeuvre-Halftermeyer, A., and Dao, T.-B.-H. (2022). Presenting an event through the description of related tweets clusters. In *ICTAI*.

Grootendorst, M. (2022). Bertopic : Neural topic modeling with a class-based tf-idf procedure. *arXiv*.

Gurcan, F., Ozyurt, O., and Cagitay, N. E. (2021). Investigation of emerging trends in the e-learning field using latent dirichlet allocation. *International Review of Research in Open and Distributed Learning*, 22(2) :1–18.

GVR, K., Shankar, R., and Pudi, V. (2010). Frequent itemset based hierarchical document clustering using wikipedia as external knowledge. In *Knowledge-Based and Intelligent Information and Engineering Systems : 14th International Conference, KES 2010, Cardiff, UK, September 8-10, 2010, Proceedings, Part II 14*, pages 11–20. Springer.

Harris, Z. S. (1954). Distributional Structure. *WORD*.

Hasan, M., Orgun, M. A., and Schwitter, R. (2019). Real-time event detection from the twitter data stream using the twitternews+ framework. *Information Processing & Management*, 56(3) :1146–1165.

Hennig, S. and Wurst, M. (2006). Incremental clustering of newsgroup articles. In *Advances in Applied Artificial Intelligence : 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006, Annecy, France, June 27-30, 2006. Proceedings 19*, pages 332–341. Springer.

Hoffman, M., Bach, F., and Blei, D. (2010). Online Learning for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems*.

Hooper, R. and Paice, C. (2005). The lancaster stemming algorithm. *University of Lancester*.

Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. *Journal for Language Technology and Computational Linguistics*, 20(1) :19–62.

Irfan, R., King, C. K., Grages, D., Ewen, S., Khan, S. U., Madani, S. A., Kolodziej, J., Wang, L., Chen, D., Rayes, A., et al. (2015). A survey on text mining in social networks. *The Knowledge Engineering Review*, 30(2) :157–170.

Jursic, M., Mozetic, I., Erjavec, T., and Lavrac, N. (2010). Lemmagen : Multilingual lemmatisation with induced ripple-down rules. *J. UCS*, 16 :1190–1214.

Kanclerz, K. and Piasecki, M. (2022). Deep neural representations for multiword expressions detection. In *ACL*.

Lau, J. H. and Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*.

Lebowitz, M. (1987). Experiments with incremental concept formation : Unimem. *Machine learning*, 2 :103–138.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27.

Li, C., Sun, A., and Datta, A. (2012). Twevent : Segment-based event detection from tweets. In *CIKM*.

Li, K. (2021). *Exploring Topic Evolution in Large Scientific Archives with Pivot Graphs*. PhD thesis, Sorbonne Université.

Liang, X., Wu, S., Li, M., and Li, Z. (2021). Unsupervised keyphrase extraction by jointly modeling local and global context. *arXiv*.

Lin, C.-Y., Cao, G., Gao, J., and Nie, J.-Y. (2006). An information-theoretic approach to automatic evaluation of summaries. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 463–470.

Lin, C.-Y. and Och, F. (2004). Looking for a few good metrics : Rouge and its evaluation. In *Ntcir workshop*.

Lin, J., Vlachos, M., Keogh, E., and Gunopulos, D. (2004). Iterative Incremental Clustering of Time Series. In Goos, G., Hartmanis, J., Van Leeuwen, J., Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K.,

and Ferrari, E., editors, *Advances in Database Technology - EDBT 2004*, volume 2992, pages 106–122. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title : Lecture Notes in Computer Science.

Liu, Q., Wang, J., Zhang, D., Yang, Y., and Wang, N. (2018). Text features extraction based on tf-idf associating semantic. In *2018 IEEE 4th international conference on computer and communications (ICCC)*, pages 2338–2343. IEEE.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics : Human language technologies*, pages 142–150.

Manchanda, P., Ansari, A., and Gupta, S. (1999). The "shopping basket" : A model for multicategory purchase incidence decisions. *Marketing science*, 18(2) :95–114.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Marchello-Nizia, C. (2018). L'évolution des langues Types et processus de changement.

Marcilio, D. S. (2019). Introduction au text mining fouille de texte.

Mayaffre, D. (2002). Les corpus réflexifs : entre architextualité et hypertextualité. *Corpus*, (1).

Maziarz, M., Szpakowicz, S., and Piasecki, M. (2015). A procedural definition of multiword lexical units. In *RANLP*.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan : Hierarchical density based clustering. *J. Open Source Softw.*

Mehrotra, R., Sanner, S., Buntine, W., and Xie, L. (2013). Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 889–892.

Meunier, R., Moudjari, L., Benamara, F., Moriceau, V., Mari, A., and Stolf, P. (2023). Classification de tweets en situation d'urgence pour la gestion de crises. In *18e Conférence en Recherche d'Information et Applications\\16e Rencontres Jeunes Chercheurs en RI\\30e Conférence sur le Traitement Automatique des Langues Naturelles\\25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, pages 204–216. ATALA.

Michalski, R. (1980). Knowledge acquisition through conceptual clustering : A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*.

Michalski, R. S. and Stepp, R. E. (1983). Automated construction of classifications : Conceptual clustering versus numerical taxonomy. *IEEE Transactions on pattern analysis and machine intelligence*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. page 9.

Miller, G. A. (1995). Wordnet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41.

Moody, C. E. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv :1605.02019*.

Newman, D., Lau, J. H., Grieser, K., and Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Human language technologies : The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 100–108.

Nyzam, V. (2021). *Résumé comparatif cross-lingue et multilingue*. PhD thesis, Paris 8.

Onik, M. M. H. and Ahmed, M. (2018). *Blockchain in the Era of Industry 4.0*, pages 259–298.

Ouali, A., Loudni, S., Lebbah, Y., Boizumault, P., Zimmermann, A., and Loukil, L. (2016). Efficiently Finding Conceptual Clustering Models with Integer Linear Programming. In *IJCAI*, page 8.

Pang, B. and Lee, L. (2004). Polarity dataset 2.0.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. (2016). The LAMBADA dataset : Word prediction requiring a broad discourse context. arXiv :1606.06031 [cs].

Paveau, M.-A. (2013). Genre de discours et technologie discursive. *Pratiques. Linguistique, littérature, didactique*, (157-158) :7–30.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Petrovic, S., Osborne, M., McCreadie, R., Macdonald, C., Ounis, I., and Shrimpton, L. (2013). Can twitter replace newswire for breaking news ? In *AAAI*, volume 7, pages 713–716.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3) :130–137.

Pritchard, J. K., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, 155(2) :945–959.

Reimers, N. and Gurevych, I. (2019). Sentence-bert : Sentence embeddings using siamese bert-networks.

Rousseeuw, P. J. (1987). Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20 :53–65.

Rowland, J. R. and Vesonder, G. T. (1987). Incremental conceptual clustering from existing databases. In *Proceedings of the 15th annual conference on Computer Science*, pages 80–87.

Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation : The basic theory. *Backpropagation : Theory, architectures and applications*, pages 1–34.

Saad, F. H., de la Iglesia, B., and Bell, D. G. (2006). A comparison of two document clustering approaches for clustering medical documents. In *DMIN*, pages 425–431. Citeseer.

Sagot, B. (2010). The lefff, a freely available and large-coverage morphological and syntactic lexicon for french. In *7th international conference on Language Resources and Evaluation (LREC 2010)*.

Sahoo, N., Callan, J., Krishnan, R., Duncan, G., and Padman, R. (2006). Incremental hierarchical clustering of text documents. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 357–366.

Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task : Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sasaki, K., Yoshikawa, T., and Furuhashi, T. (2014). Online topic model for twitter considering dynamics of user interests and topic trends. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1977–1985.

Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 298–307.

Seeman, W. and Michalski, R. (2006). The cluster3 system for goal-oriented conceptual clustering : method and preliminary results. *WIT Transactions on Information and Communication Technologies*, 37.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

STEPP, R. (1986). Conceptual clustering : Inventing goaloriented classifications of structured objects. *Machine Learning : An Artificial Intelligence Approach.*

Stevens, R., Goble, C. A., and Bechhofer, S. (2000). Ontology-based knowledge representation for bioinformatics. *Briefings in bioinformatics*, 1(4) :398–414.

Sun, J., Lapuschkin, S., Samek, W., and Binder, A. (2022). Explain and improve : Lrp-inference fine-tuning for image captioning models. *Information Fusion*, 77 :233–246.

Tarrade, L., Magué, J.-P., and Chevrot, J.-P. (2022). Detecting and categorising lexical innovations in a corpus of tweets. *Psychology of Language and Communication*, 26 :313–329.

Taslimipoor, S., Bahaadini, S., and Kochmar, E. (2020). MTLB-STRUCT @parseme 2020 : Capturing unseen multiword expressions using multi-task learning and pre-trained masked language models. In *ACL*.

Tellier, I. (2005). Introduction au TALN et 'a l'ingénierie linguistique universitée de Lille3.

Tellier, I. (2011). Introduction à la fouille de textes université de Paris 3 - Sorbonne Nouvelle.

Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., and Dyer, C. (2015). Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054.

Universalis, E. (2023). Encyclopædia Universalis.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. (2009). Evaluation methods for topic models. In *Proceedings of the 26th annual international conference on machine learning*, pages 1105–1112.

Wang, K., Ma, C., Qiao, Y., Lu, X., Hao, W., and Dong, S. (2021). A hybrid deep learning model with 1dcnn-lstm-attention networks for short-term traffic flow prediction. *Physica A : Statistical Mechanics and its Applications*, 583 :126293.

Wang, X. and McCallum, A. (2006). Topics over time : a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433.

Weng, J., Yao, Y., Leonardi, E., and Lee, F. (2011). Event Detection in Twitter. *AAAI*.

Wu, H. and Salton, G. (1981). A comparison of search term weighting : term relevance vs. inverse document frequency. In *Proceedings of the 4th annual international ACM SIGIR conference on Information storage and retrieval : theoretical issues in information retrieval*, pages 30–39.

Yan, X., Guo, J., Lan, Y., and Cheng, X. (2013). A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456.

Yin, J., Lampert, A., Cameron, M., Robinson, B., and Power, R. (2012). Using social media to enhance emergency situation awareness. *IEEE intelligent systems*, 27(06) :52–59.

Zerubavel, E. (1996). Lumping and splitting : Notes on social classification. In *Sociological Forum*, volume 11, pages 421–433. Springer.

Zhang, H. and Davidson, I. (2021). Deep Descriptive Clustering. *arXiv*.

Zhang, Q., Wang, Y., Gong, Y., and Huang, X.-J. (2016). Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 836–845.

Zhao, G., Liu, Y., Zhang, W., and Wang, Y. (2018). Tfidf based feature words extraction and topic modeling for short text. In *Proceedings of the 2018 2nd international conference on management engineering, software engineering and service sciences*, pages 188–191.

Zhao, W. X., Jiang, J., He, J., Song, Y., Achanauparp, P., Lim, E.-P., and Li, X. (2011a). Topical keyphrase extraction from twitter. In *ACL*, pages 379–388.

Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., and Li, X. (2011b). Comparing twitter and traditional media using topic models. In *ECIR*, pages 338–349. Springer.

Zheng, H.-T., Chen, H., and Gong, S.-Q. (2014). A frequent term-based multiple clustering approach for text documents. In *Web Technologies and Applications : 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings 16*, pages 602–609. Springer.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies : Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Olivier GRACIANNE

Exploration de micro-posts d'actualité : représentation, structuration et description

Résumé :
Les réseaux sociaux où sont échangés des messages courts sont devenus des sources de premier choix pour le suivi d'objets d'actualités. Les sources de ce relais d'information sont ainsi très variées, permettant de couvrir la grande majorité des aspects de ces objets. De plus, l'information est diffusée quasiment en temps réel. Traiter les données issues de ce type de plate-forme est donc devenu un objectif clef pour le monde de la recherche comme celui de l'industrie. Les présents travaux portent sur l'exploitation de ces données. L'objectif est de décrire les données issues de tels réseaux pour permettre de les rendre accessibles à un utilisateur final. Nous proposons de traiter le problème à travers la représentation, la structuration et finalement la description des données reçues. Pour pouvoir baser nos raisonnements et procédures sur la sémantique véhiculée par les données, nous procédons à leur changement de représentation. Cette étape est opérée avec un modèle d'apprentissage automatique adapté à nos besoins, Doc2Vec, produisant des représentations sémantiques raffinées. Ces vecteurs peuvent ensuite être exploités pour détecter la structuration sous-jacente de ces données. Cette étape prend la forme d'un clustering basé sur la mesure de similarité accessible dans l'espace de représentation des micro-posts captés. Ce découpage permet d'identifier les manifestations de la cible écoutée que l'on peut repérer dans les données. Nous proposons finalement d'exploiter cette partition et le contenu des messages pour identifier des composants de descriptions pertinents. Ces descripteurs, issus du corps même des messages, doivent être représentatifs de l'ensemble où ils ont été sélectionnés. Ils permettent ainsi la construction de la description de la partition. Nous introduisons d'abord à cette fin un modèle de Programmation Linéaire en Nombres Entiers, et ensuite une méthodologie basée sur le clustering et la mesure de l'impact des mots sur la position du vecteur de leur document. C'est finalement à travers l'ensemble formé par les descriptions des aspects de la cible écoutée sur les réseaux que nous proposons de la décrire.
Mots clés : Micro-posts, plongement de mot et de document, clustering, description de clustering

Exploring actuality micro-posts : representation, structuring and description

Abstract :
Social networks where short messages are exchanged have become prime sources for tracking current events. These information relay sources come from a wide variety of origins, allowing for coverage of the vast majority of the aspects of these events. Furthermore, information is shared almost in real-time. Managing data from such platforms has thus become a key objective for both the academic research and the industry. The present work focuses on the exploitation of this data. The aim is to describe data from such networks in order to make them accessible to an end user. We propose to tackle this problem through data representation, structuring, and ultimately, description. To base our reasoning and procedures on the semantics conveyed by the data, we proceed with their representation change. This step is performed using a machine learning model well suited to our needs, Doc2Vec, which produces refined semantic representations. These vectors can then be utilized to detect the underlying structure of the data. This step takes the form of clustering based on the measure of similarity accessible in the representation space of the captured micro-posts. This partitioning allows us to identify manifestations of the target event that can be found in the data. Finally, we propose to exploit this partition and the content of the messages to identify relevant description components. These descriptors, derived from the very body of the messages, must be representative of the set in which they were selected. They thus enable the construction of a description of the partition. To this end, we first introduce an Integer Linear Programming model, and then a methodology based on the obtained clustering and the measurement of the impact of words on the position of their document vector. In the end, we propose to describe the event listened on the social media through the set formed by the descriptions of its identified aspects.
Keywords : Micro-posts, word and document embedding, clustering, clustering description

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

Laboratoire d'Informatique Fondamentale d'Orléans, Batiment IIIA, Rue Léonard de Vinci, B.P. 6759, F-45067 ORLEANS Cedex 2

Atos