Analyse de sentiments sur Twitter dans un contexte faiblement supervisé

Miriam Benballa

▶ **To cite this version:**

Miriam Benballa. Analyse de sentiments sur Twitter dans un contexte faiblement supervisé. Informatique et langage [cs.CL]. Normandie Université, 2022. Français. NNT : 2022NORMIR24 . tel-04041435

HAL Id: tel-04041435
https://theses.hal.science/tel-04041435v1

Submitted on 22 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'INSA Rouen Normandie

Analyse de sentiments sur Twitter dans un contexte faiblement supervisé

Présentée et soutenue par
Miriam BENBALLA

Thèse soutenue publiquement le 18 novembre 2022 devant le jury composé de		
Vincent CLAVEAU	Chargé de recherche, CNRS, IRISA	Rapporteur
Patrice BELLOT	Professeur, LIS, Aix-Marseille Université	Rapporteur
Thierry CHARNOIS	Professeur, LIPN, Université Sorbonne Paris Nord	Examinateur
Chloé CLAVEL	Professeur, Telecom-Paris	Examinatrice
Alexandre PAUCHET	Maître de conférences HDR, LITIS, INSA Rouen Normandie	Directeur de thèse
Simon BERNARD	Maître de conférences, LITIS, Université de Rouen Normandie	Encadrant
Romain PICOT-CLÉMENTE	Head of AI, Saagie	Invité, encadrant

Thèse dirigée par Alexandre PAUCHET et co-encadrée par Simon BERNARD, laboratoire LITIS.

Saagie

INSA ROUEN NORMANDIE

MIIS

litis

Remerciements

Tout d'abord, je souhaite remercier mon directeur de thèse, Alexandre Pauchet, ainsi que mon encadrant Simon Bernard, pour leur accueil et leur accompagnement au sein du laboratoire, et pour le temps conséquent qu'ils m'ont accordée. Grâce à leurs nombreux conseils, ainsi qu'à leur franchise et leur sympathie, j'ai beaucoup appris et pu évoluer dans le monde de la recherche. Cette thèse étant réalisée dans le cadre du dispositif CIFRE, le monde industriel a également eu un impact important sur mes travaux. C'est pourquoi je souhaite également remercier Arnaud, fondateur de Saagie, ainsi que Youen, pour m'avoir accueillie, m'avoir fait confiance, et avoir encouragé le lancement de ma thèse et la création du pôle Recherche. Tout cela a également été possible grâce à Romain, mon encadrant de thèse côté entreprise, et son soutien sans faille avant, et pendant la thèse.

J'adresse aussi mes remerciements à Samia et Florian, membres de mon CSI, qui ont su être présents et rassurants dans les moments difficiles. Je tiens également à remercier les membres du LITIS pour leur accueil chaleureux, en particulier Mathieu, pour nos discussions et débats sur les jeux vidéos qui m'ont fait sortir de ma coquille, ainsi que Sandra et Brigitte, toujours disponibles et prêtes à aider.

Un grand merci à Saagie, et en particulier à l'équipe de Rouen, qui m'a chaleureusement accueillie. Ils m'ont permis d'évoluer ces dernières années, que ce soit en développant mes réflexes grâce à l'esquive des balles de Kevin, ma rapidité d'exécution après des heures passées sur Guitar Hero avec Guillaume, ou encore ma patience et mon sens de l'organisation grâce aux évènements mis en place en compagnie des collègues de la Team-Happy. Merci pour toutes ces soirées organisées (et ainsi que pour tous ceux qui, heureusement, n'ont jamais vu le jour), et merci aux habitués qui n'ont pas eu peur d'y participer : Alan, Anne, Audrey, Bilal, Celestin, Charlotte, Cyrielle, Florence, Guillaume, Héloïse, Hugo L. et Hugo P., Julien, Kevin Leshan, Pierre, Romain, Tony, ou encore Victor.

Enfin, je remercie également mes parents, mes frères et sœurs, ainsi que Tony pour leur soutien et leur patience, en particulier cette dernière année.

Abstract

This manuscript describes the research carried out in the scope of the CIFRE thesis conducted in partnership between LITIS and Saagie. In the different chapters, we focus on the task of sentiment analysis, using the current state of the art models, Transformer models. More specifically, we focus on BERT, that inspired many of the Transformer-based models.

We first define a list of guidelines in order to perform sentiment analysis with these state-of-the-art models, while having access to limited resources. Indeed, Transformer models are very greedy in terms of data resources, memory, and computing power which are very expensive, and therefore difficult to access. Thus, we recommend the use of a specialized model when it is available, such as BERTweet which is funded on the BERT-base architecture. In cases where the selected model does not have a specialized equivalent, we recommend the use of fine-tuning to adapt the generic model to tweets. Finally, the model may be susceptible to catastrophic forgetting during its fine-tuning. This can be avoided by using a low learning rate, as well as by freezing part of the model layers.

In a second step, we seek to improve sentiment analysis by performing different combinations of the following models : BERTweet latent space fine-tuned on this task, Emojional emoji representation, and BERT-base latent space fine-tuned on sarcasm detection. It turns out that the emoji representation significantly improves sentiment classification, but the addition of the sarcasm model inhibits this improvement. In our further study, we hypothesized two reasons for this limitation : the sarcasm information contradicts the information from the other models, or the combination method used does not allow a proper integration of the sarcasm information. These hypotheses will be tested outside the scope of this thesis.

Finally, we present the SAPhIRS project developed in partnership between Saagie, LITIS and Airbus Defence and Space, in which this thesis is included. This project aims to detect radicalization, influencers and communities on Twitter, on a dataset created by an external company (ELDA). Each of the modules created within the scope of this project integrates a processing chain (pipeline) that is put into production on the Saagie platform. Moreover, while waiting for the corpus created by ELDA, we participated in the hate detection task of the SemEval 2019 competition. Indeed, this task remains close to our radicalization theme.

Résumé

Ce manuscrit présente les travaux effectués dans le cadre de la thèse CIFRE réalisée en partenariat entre le LITIS et Saagie. Au cours des différents chapitres, nous nous intéressons à la tâche d'analyse de sentiments, en utilisant les modèles actuellement à l'état de l'art, les modèles fondés sur le Transformer. Plus particulièrement, nous nous concentrons sur le modèle BERT, modèle ayant inspiré bon nombre des modèles fondés sur le Transformer.

Nous commençons tout d'abord par définir une liste de recommandations à prendre en compte afin de faire de l'analyse de sentiments avec ces modèles à l'état de l'art, tout en ayant accès uniquement à des ressources limitées. En effet, les modèles Transformer sont très gourmands en ressources données, mémoire, et puissance de calcul. Ces ressources sont très coûteuses, et donc difficile d'accès. Ainsi, nous recommandons l'utilisation d'un modèle spécialisé lorsque celui-ci est disponible, comme BERTweet reprenant l'architecture de BERT-base et spécialisé dans le traitement des tweets. Dans les cas où le modèle sélectionné ne possède pas d'équivalent spécialisé, nous recommandons l'utilisation du fine-tuning afin d'adapter le modèle générique aux spécificités de la tâche. Enfin, lors de son fine-tuning, le modèle peut être sensible à l'oubli catastrophique. Ceci peut être évité grâce à l'utilisation d'un taux d'apprentissage faible, ainsi qu'en gelant une partie des couches du modèle.

Dans un second temps, nous cherchons à améliorer l'analyse de sentiments en réalisant différentes combinaisons des modèles suivants : l'espace latent de BERTweet affiné sur cette tâche, la représentation d'emojis Emojional, et l'espace latent de BERT-base affiné sur la détection de sarcasme. Il s'avère que la représentation d'emojis améliore grandement la classification de sentiments, mais que l'ajout du modèle de sarcasme freine cette amélioration. En poussant notre étude, nous avons émis deux hypothèses concernant ce frein : les informations de sarcasme entrent en contradiction avec les informations des autres modèles, ou la méthode de combinaison utilisée ne permet pas une bonne intégration de l'information de sarcasme. Ces hypothèses seront vérifiées en dehors du cadre de cette thèse.

Enfin, nous présentons le projet SAPhIRS réalisé en partenariat entre Saagie, le LITIS et Airbus Defence and Space, et dans lequel s'inscrit cette thèse. Ce projet a pour objectif la détection de radicalisation, d'influenceurs et de communautés sur Twitter, sur un jeu de données constitué par une société externe (ELDA). Chacun des modules créés dans le cadre de ce projet intègre une chaîne de traitement (pipeline) mis en pro-

duction sur la plateforme Saagie. De plus, dans l'attente du corpus collecté par ELDA, nous avons participé à la tâche de détection de haine de la compétition SemEval 2019. En effet, cette tâche reste proche de notre thématique de radicalisation.

Table des matières

Liste des tableaux

Table des figures

Glossaire

Apprentissage auto-supervisé Méthode d'apprentissage non supervisé, qui permet d'apprendre un modèle sur des données non annotées. La technique générale de l'apprentissage auto-supervisé consiste à prédire toute partie (ou propriété) non observée ou cachée de l'entrée à partir de toute partie observée ou non cachée de l'entrée. Plus de détails en partie 2.3.3 Meta AI.

Apprentissage non supervisé Apprentissage automatique en utilisant des données non annotées.

Apprentissage par batch Mode de traitement des données suivant lequel les données d'apprentissage sont groupées par lots. Datafranca.

Apprentissage supervisé Apprentissage automatique en utilisant des données annotées.

Arabizi Alphabet de tchat arabe.

Attention multi-head Utilisation du principe de l'attention en utilisant plusieurs têtes, ce qui permet d'exécuter le mécanisme d'attention plusieurs fois en parallèle. Plus de détails en partie 2.3.3.

Auto-attention Application du mécanisme de l'attention sur une unique séquence au lieu de deux). Ainsi, les relations entre les différents tokens d'une même séquence seront calculées. Plus de détails en partie 2.2.1.3 et 2.3.3.

BERT *Bidirectional Encoder Representations from Transformers*, modèle de langage fondé sur le Transformer, créé par Google en 2019 [33]. Plus de détails en partie 2.3.3.

BERTweet Architecture de BERT-base entraîné sur des données provenant de Twitter [104].

CBOW Méthode d'apprentissage d'un *embedding* consistant à prédire un mot étant donné les mots de son contexte.

CNN *Convolutionnal Neural Network*, type de réseau de neurones artificiels à propagation avant (*feed-forward*), dans lequel les connexions entre les neurones sont inspirées par le cortex visuel des animaux [79]. Plus de détails en partie 2.3.2.

CRF *Conditional random fields*, Champs aléatoires conditionnels en français, il s'agit d'une classe de modèles statistiques utilisés en reconnaissance des formes et plus généralement en apprentissage statistique.

Disparition du gradient Se produit lorsque le gradient diminue jusqu'à tendre vers 0. Les poids ne sont alors plus mis à jour, et la descente de gradient ne converge jamais vers l'optimum.

Démojisation Conversion des emojis en leur texte (:thumbsup: au lieu de 👍).

Embedding Parfois traduit en français « plongement », mais le terme anglais est plus courant, il s'agit d'un modèle résultant de l'opération mathématique qui permet de passer d'une représentation vectorielle à une nouvelle représentation où les objets similaires possèdent des vecteurs correspondants qui sont proches dans l'espace vectoriel où sont définis ces vecteurs (typiquement par un algorithme qui réduit la dimension de la représentation afin de rapprocher les objets similaires et éloigner les objets différents). Plus de détails en partie 2.2.1.2 Datafranca.

Emoji Pictogrammes représentant un sentiment, une émotion, un objet ou même une idée.

Emoji2vec *Embedding* d'emojis adapté du Word2vec, appris avec le skip-gram de Google. Les données Twitter collecté pour l'apprentissage de cette représentation a permis d'obtenir 6088 descriptions de 1661 emojis [35].

Emojional *Embedding* d'emojis fondé sur Emoji2vec, appris avec le skip-gram de Google. Les données Twitter collecté pour l'apprentissage de cette représentation a permis d'obtenir 10,854 descriptions, permettant de représenter 1816 emojis [8].

Explosion du gradient Se produit lorsque le gradient augmente de façon considérable au cours de la rétropropagation. Les mises à jour de poids deviennent alors très importantes et dévient la descente du gradient.

F1-score Métrique de classification qui mesure la capacité d'un modèle à bien prédire les individus positifs, tant en termes de precision (taux de prédictions positives correctes) qu'en termes de recall (taux de positifs correctement prédits). Kobia.

Fine-tuning Affiner un modèle pour l'adapter à un nouveau jeu de données ou à une nouvelle tâche, en réalisant quelques epochs d'entraînement sur cette tâche. Plus de détails en partie 3.1.1.

Fonction de perte Fonction qui évalue l'écart entre les prédictions réalisées par le réseau de neurones et les valeurs réelles des observations.

GloVe Global Vector, il s'agit d'un *embedding* de mots, publié par (Pennington et al., 2014) entraîné sur les entrées non nulles d'une matrice de co-occurrence indiquant la fréquence de co-occurrence des mots entre eux dans un corpus donné [115].

GRU *Gated Recurrent Unit*, adaptation plus simple des LSTM [22]. Plus de détails en partie 2.3.2.

Hashtag « Mot dièse », mots précédés du symbole « # » qui correspondent à une thématique attribuée au message.

Lemmatisation Retrait de la flexion des mots. La flexion correspond à la déclinaison pour les noms (chat/chats), adjectifs (petit/petite), pronoms (mien/miennes) ou à la conjugaison pour les verbes (écrire/écrivons).

LSTM *Long-Short Term Memory*, réseau de neurones artificiels présentant des connexions récurrentes, il s'agit d'une modification des RNN permettant de corriger les problèmes de disparition et d'explosion du gradient [57]. Plus de détails en partie 2.3.2.

Mention d'utilisateur Sur Twitter, il s'agit du nom d'utilisateur précédé du symbole « @ ».

MLM Mask Language Modelling, tâche d'entraînement du modèle BERT, dans laquelle le modèle tente de prédire un token qui a été masqué [33]. Plus de détails en partie 2.3.3.

N-gramme Séquence de n mots adjacents contenus dans un corpus, utilisé pour la représentation du texte.

NB Classifieur probabiliste fondé sur le théorème de Bayes, permet de classifier les textes en fonction des mots qu'ils contiennent, et de leur probabilité d'appartenance à l'une des classes. Plus de détails en partie 2.3.1.

NSP Next Sentence Prediction, tâche d'entraînement du modèle BERT, le but étant de prédire si une séquence B suit une séquence A [33]. Plus de détails en partie 2.3.3.

Opinion Quintuplet (o, a, p, e, t) (objet de l'opinion, aspect de cet objet, polarité de l'opinion, émetteur de l'opinion, moment d'émission de l'opinion) [70, 86]. Plus de détails en partie 2.1.1.

Oubli catastrophique Tendance que peut avoir un modèle durant le *fine-tuning* à oublier les connaissances qu'il a acquises lors de son pré-entraînement [97]. Plus de détails en partie 3.1.1.

POS tag *Part-of-speech* tag, il s'agit d'une étiquette indiquant la nature d'un mot d'un point de vue grammatical (nom, verbe, adjectif, etc..

Poursuite du pré-apprentissage Poursuivre le pré-apprentissage d'un modèle en utilisant les tâche de son pré-entraînement. Plus de détails en partie 3.1.2.

Racinisation Transformation d'un mot en sa racine.

Radicalisation Processus d'adoption d'une croyance extrémiste incluant la volonté d'utiliser, de soutenir ou de faciliter la violence comme méthode de changement de la société hrefhttp://radicalisation.fr/radicalisation_definition.phpradicalisation.fr.

Représentation binaire d'un texte Représentation du texte à l'aide d'un vecteur $T = \{t_1, t_2, ..., t_n\}$, dans lequel chaque mot t_i est représenté par 1 s'il est contenu dans le document, et 0 s'il est absent.

Retweet Message ayant été reposté. Les retweets sont précédés du tag « RT ».

RNN *Recurrent Neural Networks*, réseau de neurones artificiels présentant des connexions récurrentes [129]. Plus de détails en partie 2.3.2.

Rétropropagation du gradient Algorithme d'optimisation permettant d'ajuster les paramètres d'un réseau de neurones en apprentissage. Le principe est de mettre

à jour les poids des neurones de la dernière à la première selon les prédictions réalisées et les valeurs réelles des observations afin de converger vers un minimum global, qui correspond alors à une configuration optimale des poids..

Sac de mots Méthode de représentation du texte, permettant d'associer un vecteur de réels à chaque document. Plus de détails en partie 2.2.1.1.

Sentiment Expression d'une subjectivité (ou de l'absence de subjectivité) dans un texte, sous la forme d'une valence ou une étiquette positive. Plus de détails en partie 2.1.1.

Skip-gram Méthode d'apprentissage d'un *embedding* consistant à prédire les mots du contexte étant donné un mot en entrée.

Slanted triangular learning rate Schéma de taux d'apprentissage qui augmente d'abord de manière linéaire le taux d'apprentissage puis le diminue également de manière linéaire [59]. Plus de détails en partie 3.1.1.

Stop words Mots « vide de sens » très présents dans les textes (pronoms, déterminants, etc.).

SVM Algorithme de classification dont l'objectif est de trouver un hyperplan permettant de séparer les classes dans l'espace de projection grâce au vocabulaire vectorisé des tweets, et ce, de la manière la plus optimale possible. Plus de détails en partie 2.3.1.

TAL Traitement automatique du langage, il s'agit d'une discipline visant à créer des outils de traitement de la langue de manière automatique.

Taux d'apprentissage Taux contrôlant la vitesse à laquelle les pondérations sont ajustées. Il s'agit d'un hyper-paramètre utilisé dans beaucoup de méthode d'apprentissage.

TF.IDF *Term Frequency Inverse Document Frequency*, représentation du texte utilisant la fréquence et permettant de donner plus d'importance aux mots peu présents dans le corpus grâce à leur pondération par l'IDF qui correspond au score de « rareté » d'un token t (l'importance d'un mot dans un document, par rapport au corpus complet de documents).

Token Instance d'une séquence de caractères dans un document, regroupés comme une unité sémantique utile pour le traitement Stanford NLP Group.

Transfer learning Apprentissage par transfert en français, désigne l'ensemble des méthodes qui permettent de transférer les connaissances acquises à partir de la résolution de problèmes donnés pour traiter un autre problème. Plus de détails en partie 3.1.1.

Twitter Service de microblogging créé aux États-Unis en 2006, qui permet aux utilisateurs d'envoyer et de recevoir des messages brefs, appelés Tweets Larousse.

URL Lien hypertexte.

VADER *Valence Aware Dictionary and sEntiment Reasoner*, VADER est un outil d'analyse de sentiment fondé sur des règles et du lexique spécialisé dans les sentiments exprimés dans les réseaux sociaux Github.

Word2vec *Embedding* de mots publié par (Mikolov et al., 2013) et appris avec le CBOW [98]. Une seconde version fut publié la même année par (Mikolov et al., 2013) appris cette fois avec le skip-gram [99].

Chapitre 1

Introduction

1.1 Contexte de la thèse

L'analyse de sentiments est une tâche prépondérante du Traitement Automatique du Langage (TAL). Depuis longtemps, la littérature s'intéresse à l'extraction et à l'analyse de la polarité des sentiments présents dans le texte. Par exemple, il est fréquent de classifier la polarité des commentaires des produits présents sur Amazon [14, 51, 101, 122]. En effet, pour ce genre de corpus, la présence d'une note en plus du commentaire permet d'avoir une information sur la polarité de ce dernier, ce qui en fait des données propices à la classification de sentiments. L'analyse de revues de film est un autre exemple de tâche courante en TAL [44, 95, 148, 166]. Encore une fois, ce type de corpus a l'avantage de contenir des notes clarifiant la polarité des commentaires.

Afin de réaliser l'analyse de sentiments, la meilleure solution est l'utilisation des modèles Transformer, actuellement à l'état de l'art. Effectivement, depuis la publication du Transformer en 2017 dans [151], la majorité des modèles récents utilisent ce mécanisme. Le Transformer a pour principe central l'utilisation de l'auto-attention, c'est-à-dire la mesure des relations entre les différents *tokens* d'une même séquence. Ainsi, dans le cadre du TAL, l'auto-attention permet de trouver les relations entre les différentes parties d'une même phrase. BERT [33] et GPT [118] sont les pionniers parmi les modèles utilisant le bloc Transformer, et ont inspiré les modèles Transformer suivants.

Les modèles fondés sur le bloc Transformer ont l'avantage d'être très performants. Leur apprentissage est effectué grâce au principe de l'auto-supervision. Grâce à cela, les modèles sont pré-appris sur de très grandes quantités de données non annotées, par le biais de différentes tâches selon le modèle et en utilisant de nombreux GPU. Ces aspects permettent d'obtenir des modèles génériques pré-entraînés puissants, utilisables après *fine-tuning*[1] [128]. Cependant, la quantité de données utilisées et surtout la puissance

1. Réglage fin ou affinage en français, consiste à adapter les poids d'un modèle pré-entraîné sur des données sources en utilisant un jeu de données cible grâce à de l'entraînement supplémentaire, nous conservons ici le terme anglais car plus répandu

de calcul et la mémoire indispensables à ces modèles Transformer compliquent leur apprentissage.

En ce qui concerne la mémoire et la puissance de calcul, ce sont des ressources particulièrement chères et difficilement accessibles dans de telles quantités. Par exemple, l'entraînement du modèle GPT-3 [17] a nécessité l'équivalent de 355 GPU-années, soit environ 4.6 millions de dollars. Ainsi, ce type d'apprentissage n'est réalisable que par les grandes entreprises comme OpenAI [2] ou Google [3], et difficilement possible pour des utilisateurs à plus petite échelle. Concernant la quantité de données nécessaires au pré-entraînement, la plupart des modèles utilisent d'énormes corpus de données génériques. Par exemple, BERT a été pré-entraîné en utilisant le Wikipédia en anglais (2 500 millions de mots) et le BooksCorpus [176] (800 millions de mots). Si cette quantité de données est facilement disponible, les traiter sans la puissance de calcul ou la mémoire adéquate est extrêmement long.

Puisqu'ils sont entraînés sur des corpus génériques, ces modèles ne sont pas adaptés aux données spécifiques (c'est-à-dire traitant une thématique ou un média particulier, comme Twitter) sans étape de *fine-tuning*. Ainsi, dans le cas de l'analyse de sentiments sur des tweets, il est important d'adapter le modèle au langage et aux codes de Twitter [4]. En effet, ce réseau social contient un vocabulaire spécifique, comme des hashtags (ou « mot-dièse »), des mentions d'utilisateurs, des abréviations, des fautes d'orthographe, des emojis, un langage familier, etc. [46, 75]. Ces spécificités doivent être prises en compte, afin d'analyser les tweets de la meilleure manière possible. En effectuant de l'analyse de tweets avec un modèle générique, le vocabulaire spécifique à Twitter n'a pas été vu lors du pré-apprentissage.

Nous effectuons de l'analyse de sentiments avec des données issues du réseau social Twitter [47, 70, 86, 111]. Nous proposons donc dans ce manuscrit des solutions aux différents problèmes identifiés pour cette tâche, en utilisant des modèles à l'état de l'art. Pour cela, nous nous intéressons au manque de ressources pour l'utilisation des modèles Transformer et à la prise en compte des spécificités de Twitter.

De plus, cette thèse étant réalisée dans le cadre du projet SAPhIRS, décrit en Chapitre 5, nous travaillons sur des thématiques sensibles comme la haine ou la radicalisation. En effet, le réseau social Twitter est également utilisé dans le cadre de la propagation de la haine et de la violence. Malgré une modération permettant la suppression des messages haineux, racistes, homophobes ou encore faisant l'apologie du terrorisme, le réseau a notamment servi au recrutement dans le cadre de la guerre en Syrie et des attentats ayant eu lieu dans le monde ces dernières années.

Dans la partie suivante, nous décrivons le projet SAPhIRS, et nous montrons comment cette thèse s'inscrit dans ce projet.

2. OpenAI
3. Google
4. Twitter

1.2 Projet SAPhIRS

Dans le cadre de cette thèse, nous faisons de l'analyse de sentiments dans les données Twitter. D'une part, cela est dû aux sujets traités. En effet, cette thèse est lié au projet SAPhIRS (Système pour l'Analyse de la Propagation d'Information dans les Réseaux Sociaux), décrit dans le Chapitre 5. Ce projet est réalisé en partenariat entre Saagie, le LITIS et Airbus Defence and Space. Il traite de la détection de radicalisation, d'influenceurs et de communautés dans les réseaux sociaux et plus particulièrement sur Twitter. Puisque l'objectif du projet SAPhIRS est la détection de radicalisation et d'influenceurs, un réseau social comme Twitter est idéal pour détecter les utilisateurs ayant des interactions et formant des communautés. La nature de Twitter favorisant la propagation d'idées, quelles qu'elles soient, il s'agit d'un support adéquat pour se procurer des données de ces thématiques.

D'autre part, la simplicité d'accès des données Twitter nous permet d'avoir à disposition une énorme quantité de données, puisque les tweets ainsi que leurs méta-données peuvent être récupérés très facilement grâce à l'API officielle. Cependant, deux problèmes se posent. Dans un premier temps, ces données ne sont pas annotées, et compte tenu de nos thématiques de recherche, leur annotation est compliquée. Dans un second temps, dû encore une fois au thème de la haine et de la radicalisation, les tweets sont plus compliqués à obtenir car rapidement supprimés des réseaux sociaux.

En traitant des thématiques comme la haine et la radicalisation, nous avons souhaité travailler sur des sujets d'actualités tout en étant utiles à la lutte contre le cyber-harcèlement et la propagation d'idéologies radicales. Sur le plan civil, l'analyse de sentiments permet également à Saagie de fournir à ses clients une analyse des réseaux sociaux. Les différentes contributions réalisées pour ce projet et présentées dans ce manuscrit sont donc mises en production, et publiées sur la plateforme Big Data Saagie. Dans la partie suivante, nous voyons quelles sont ces différentes contributions.

1.3 Contributions

De nombreuses études s'intéressent à l'analyse de sentiments dans des données issues de Twitter. Dans ce manuscrit, nous décrivons en détail nos axes de recherches ainsi que nos contributions à l'analyse de sentiments.

Dans un premier temps nous nous sommes orientés vers la recherche d'une solution afin d'utiliser des modèles actuellement à l'état de l'art, les modèles Transformer. Comme nous l'avons vu précédemment, ces modèles nécessitent de grandes quantités de ressources (données, mémoire et puissance de calcul), difficilement accessible à tous. Dans ce manuscrit, nous présentons donc des lignes directrices permettant de contourner cette nécessité, et ainsi d'utiliser les modèles Transformer pour la classification de sentiments.

Dans un second temps, en réutilisant nos précédentes recommandations, nous proposons d'essayer d'améliorer la classification du sentiment en intégrant une représentation d'emojis ainsi que des caractéristiques de sarcasme. Nous étudions donc la combinaison

de ces trois informations en utilisant la concaténation, la somme ainsi que la moyenne. Ceci nous a permis de mettre en avant le fait que, bien que le sarcasme soit une composante importante à la détection de sentiments, cette information gêne le modèle lorsqu'elle est combinée à des caractéristiques de sentiments ou d'emojis.

Dans la partie suivante, nous verrons comment les différents chapitres de ce manuscrit sont organisés.

1.4 Organisation

Ce document s'organise de la manière suivante : dans le Chapitre 2, nous nous concentrons sur notre tâche d'analyse de sentiments. Nous commençons par formaliser notre problématique, avant d'étudier les différents outils qui nous permettent de faire de l'analyse de tweets. Nous proposons ensuite un panorama des différentes approches de classification de la littérature, ainsi que les méthodes de pré-traitements et de représentation des tweets.

Le Chapitre 3 aborde l'un des obstacles majeurs aujourd'hui au traitement de la langue avec des méthodes à l'état de l'art : la grande quantité de ressources en données, mémoire et surtout en puissance de calcul qui sont nécessaires pour en exploiter le potentiel. Pour autant, il existe des solutions scientifiques et pratiques. Nous proposons dans ce chapitre de les lister, de les analyser, et d'en tirer des recommandations pour l'analyse de sentiments dans les tweets avec des ressources limitées.

Dans le Chapitre 4, nous étudions si l'ajout d'information de sarcasme et d'emojis permet d'améliorer la classification de sentiment. En restant dans notre cadre de ressources limitées en puissance de calcul, nous voyons comment ces différentes informations peuvent être fusionnées, ainsi que l'impact qu'elles peuvent avoir lorsqu'elles sont utilisées seules avec un détecteur de sentiments, ou combinées.

Le Chapitre 5 est dédié à l'industrialisation et à la mise en production d'algorithmes d'apprentissage profond. Nous y décrivons le projet SAPhIRS, auquel cette thèse est liée, qui traite de radicalisation et de détection d'influenceurs sur Twitter. Dans le cadre de ce projet, l'équipe Recherche de Saagie a également participé à la tâche 5 de la compétition SemEval 2019, et réalisé de la détection de haine. Ce chapitre décrit également l'approche utilisée pour cette tâche.

Enfin, nous terminerons ce manuscrit avec un récapitulatif de nos contributions. Nous conclurons sur nos différents apports à l'analyse de sentiments sur Twitter, et nous proposerons également des perspectives et des pistes d'améliorations des points traités dans les différents chapitres.

Chapitre 2

Analyse de sentiments sur Twitter

Résumé

Dans ce chapitre nous réalisons un état de l'art des différentes méthodes traditionnellement utilisées dans la résolution de notre problématique. Tout d'abord nous faisons la distinction entre les tâches d'analyse de sentiment et de détection d'opinion, en définissant également chacun des éléments qui composent un sentiment et une opinion. Nous nous penchons ensuite sur les différents traitements appliqués au texte pour l'analyse de tweets, avant d'étudier les algorithmes principalement utilisés pour l'analyse de sentiment. Nous commençons tout d'abord par faire un état des lieux des principaux réseaux de neurones utilisés en traitement du langage, et ce afin de sélectionner le plus adapté à notre tâche. Nous présentons donc les différents types d'architectures classiques de réseaux de neurones utilisés en traitement du langage, c'est-à-dire les RNN, LSTM, GRU ainsi que les CNN adaptés pour le traitement du texte, puis nous nous penchons vers les modèles type Transformer fondés sur de l'attention, comme GPT, BERT, et leur différentes déclinaisons. Ceci nous permet de sélectionner BERT comme étant l'architecture la plus appropriée pour nos travaux. Nous voyons donc que, bien qu'il soit possible d'utiliser des techniques d'apprentissage automatique classiques pour la classification de tweets, les réseaux de neurones, et plus précisément les architectures fondées sur les Transformer, sont aujourd'hui majoritairement utilisées.

Sommaire

Analyse de sentiments sur Twitter

Les réseaux sociaux sont devenus de plus en plus présents dans la vie des gens, permettant d'exprimer des sentiments et des opinions sur des sujets publics ou politiques, ou même sur la vie quotidienne. Pouvoir extraire le sentiment et l'opinion des tweets est donc devenu essentiel puisque tout le monde publie son opinion envers presque tout. Les tweets liés aux marques sont particulièrement utiles pour les entreprises qui souhaitent évaluer leur réputation, améliorer leurs produits, ou étudier les besoins du marché et leurs concurrents. (Ghiassi et al., 2016) ont par exemple développé une approche qui permet d'effectuer une analyse des sentiments sur Twitter et de classer les tweets sur une échelle en cinq points [43].

De ce fait, de nombreuses études se penchent sur la détection de sentiments et d'opinions contenus dans des tweets [47, 70, 86, 111]. Cependant, la plupart de ces études parlent d'analyse de sentiments ou d'opinions sans faire de distinction. Il s'agit pourtant de deux tâches proches mais pas totalement identiques [70, 86].

Si les approches traditionnelles pour l'apprentissage automatique et les réseaux de neurones récurrents et convolutifs pour le traitement du texte ont chacun leur tour permis d'obtenir les meilleures performances pour la classification de tweets, ce sont actuellement les modèles fondés sur le Transformer qui sont à l'état de l'art sur cette tâche. En effet, depuis l'apparition du bloc Transformer [151] (expliqué sous-section 2.3.3) en 2017, celui-ci est présent dans la majorité des modèles publiés récemment et pour l'analyse de texte.

Grâce à leurs nombreuses couches permettant une meilleure représentation du texte, les modèles Transformer capturent des caractéristiques à la fois sémantiques et syntaxiques, surpassant ainsi les représentations « sac de mots » et les *embeddings*[1] (ces notions sont expliquées dans la sous-section 2.2.1).

Dans ce chapitre, nous commençons donc par faire la distinction entre les tâches d'analyse de sentiments, et de détection d'opinion afin d'éclaircir ces deux notions. Nous voyons ensuite différentes manières de représenter un texte sous la forme d'un vecteur ainsi que des méthodes de traitement du texte, et en particulier des tweets. Nous faisons ensuite un point sur les différentes approches permettant d'effectuer de l'analyse de sentiments. Enfin, nous terminons par présenter les différents jeux de données utilisés dans l'ensemble de nos expérimentations.

1. Parfois traduit en français « plongement », mais le terme anglais est plus courant, il s'agit d'un modèle résultant de l'opération mathématique qui permet de passer d'une représentation vectorielle à une nouvelle représentation où les objets similaires possèdent des vecteurs correspondants qui sont proches dans l'espace vectoriel où sont définis ces vecteurs (typiquement par un algorithme qui réduit la dimension de la représentation afin de rapprocher les objets similaires et d'éloigner les objets différents) Datafranca

2.1 Formalisation du problème

Analyse de sentiments, reconnaissance d'opinion et détection de posture sont trois tâches similaires mais pourtant différentes en Traitement Automatique du Langage (TAL) [70, 86]. Il est donc nécessaire de rappeler la distinction entre ces trois tâches souvent confondues. Cette section est consacrée à la clarification de notre problématique. Dans un premier temps, nous définissons chacune des tâches afin de préciser les différents termes utilisés dans la littérature, concernant la reconnaissance d'opinion, l'analyse de sentiments et la détection de posture. Nous prenons ensuite position parmi ces tâches et définitions. Enfin, nous mettons en lumière les particularités liées à notre tâche et à notre thématique.

2.1.1 Sentiment et Opinion

Un sentiment est l'expression d'une subjectivité (ou de l'absence de subjectivité) dans un texte, sous la forme d'une valence ou une étiquette positive (exemple 1), négative (exemple 2) ou hors contexte (exemple 3).

> **Exemple 1** I'm about to eat four hot dogs and watch Miss USA. Happy Sunday.

> **Exemple 2** Bad Blood may have the absolute worst lyricism I've ever heard in my life (regarding Taylors parts) #UsedtobeMadLove #Really-DeepCut

> **Exemple 3** Trump to skip Miss USA pageant : Donald Trump won't be at the Miss USA Pageant taking place in Baton Rouge, Louisiana, on Sunday night...

Une opinion est une notion plus complexe, car définie au travers de cinq entités. Cela consiste à extraire et analyser l'opinion O d'une entité envers une autre entité. Cela signifie que l'entrée est un texte (un document, une phrase, ou dans notre cas, un tweet) ou une séquence de textes. En ce qui concerne la sortie obtenue, il s'agit de l'opinion O qui peut être représentée par le quintuplet (o, a, p, e, t) [70, 86]. Soit :

o l'objet de l'opinion,

a l'aspect de cet objet,

p la polarité de l'opinion,

e l'émetteur de l'opinion,

t le moment d'émission de l'opinion.

Pour l'exemple 4, o serait le *new Moto G*, l'aspect a serait son prix, et la polarité p serait positive. Concernant e et t, ils ne sont pas présents dans le texte, cependant s'agissant d'un tweet, l'émetteur e peut être considéré comme étant l'auteur du tweet, et t le moment de publication du tweet.

Exemple 4 The new Moto G may be the best budget phone... again. http://t.co/GXy3TDlRpC #News #TechNews

Au sens strict de ces deux notions, le sentiment est la polarité d'un texte indépendamment de tout objet et/ou d'un émetteur, alors que l'opinion est définie au travers des cinq notions ci-dessus. Dans le cadre de l'analyse de sentiments, il est souvent fait l'hypothèse simplificatrice que chacun des documents d'un jeu de données est fondé sur un même objet d'opinion, unique et éventuellement déterminé par les mots-clés utilisés lors de la collecte du corpus [112].

2.1.2 Détection de posture, analyse de sentiments fondée sur l'aspect & analyse de sentiments fondé sur la cible

Si l'analyse de sentiments permet de prédire une polarité et la reconnaissance d'opinion permet de prédire un quintuplet, la détection de posture (*Stance detection*) se positionne entre les deux. En effet, l'objectif ici est de déterminer une polarité vis-à-vis d'un objet ciblé, faisant ou non partie du document étudié. Contrairement à l'analyse de sentiments, nous ne faisons donc pas l'hypothèse que le jeu de données a été collecté sur une thématique particulière, ou encore que chaque document a un objet unique, puisque cet ou ces objets restent à déterminer [133].

Il est possible de considérer que deux sous-problèmes d'analyse de sentiments se rapprochent beaucoup de la tâche de détection de posture [74], ainsi que de la reconnaissance d'opinion. Il s'agit en effet de l'analyse de sentiments fondé sur la cible (*Target-based sentiment analysis*) ainsi que de l'analyse de sentiments fondée sur l'aspect (*Aspect-based sentiment analysis*).

Analyse de sentiments fondée sur la cible
L'objectif de cette tâche est de déterminer la polarité ainsi que la cible, c'est-à-dire l'objet o d'un document [74]. Selon (Ebrahimi et al., 2016) , les principales différences avec la détection de posture, sont dans un premier temps liées à la cible, qui peut ne pas être explicitement présente dans le document pour la détection de posture [34]. Dans un second temps, la cible de la détection de posture n'est pas forcément la cible du sentiment présent dans le document. Enfin, si dans le cas de l'analyse de sentiments la cible est habituellement une entité ou une caractéristique, elle peut être un événement en détection de posture. Par exemple, dans le cadre d'une collecte de tweets réalisée sur la thématique du second tour des élections présidentielles 2022 en France, seraient considérés comme favorable à E. Macron des tweets contre M. Le Pen. Ainsi, la cible (E. Macron) n'est ni explicitement contenue dans le tweet, ni la cible du sentiment du texte.

Analyse de sentiments fondée sur l'aspect
Dans le cas de l'analyse de sentiments fondée sur l'aspect, le but est de prédire une polarité, mais également une cible (un objet o) et une caractéristique de cet objet (un aspect a) [74]. Cette tâche se rapproche donc de la reconnaissance d'opinion, car la sortie n'est pas un élément unique, mais un triplet (o, a, p). Ainsi, dans le cadre

d'une campagne menée sur les Iphone, la sortie à prédire contiendrait le modèle du téléphone, une caractéristique de ce modèle, ainsi que la polarité du sentiment sur cette caractéristique (par exemple une polarité négative sur la batterie de l'Iphone 12).

2.1.3 Positionnement

Au commencement de cette thèse, notre objectif était de réaliser de la reconnaissance de changement d'opinion dans des tweets, en émettant les hypothèses simplificatrices suivantes :

l'objet o est connu et correspond à l'objet du tweet,

l'aspect a est connu et correspond à l'objet lui-même,

l'émetteur e est connu et correspond à l'émetteur du tweet,

le temps t est connu et correspond au moment d'émission du tweet.

L'idée était de réaliser cette reconnaissance de changement d'opinion dans une séquence de tweets d'un même auteur, et ce concernant un objet prédéterminé, en détectant les changements de polarité au fil de la séquence. L'entraînement d'un modèle sur la reconnaissance d'opinion devait être réalisé grâce à un jeu de données collectées et annotées spécialement dans le cadre du projet SAPhIRS (projet dans lequel s'inscrit cette thèse) par un prestataire extérieur.

Malheureusement, la constitution de ce corpus a pris du retard pendant nos travaux. De plus, la constitution de séquences d'un même auteur sur les thématiques traitées par le projet SAPhIRS (haine, radicalisation) fut impossible. Puisque nous n'avons pas eu accès à ces séquences, et qu'il n'existe pas à notre connaissance de jeux de données similaires, publiquement accessibles, nous nous sommes consacrés à la détection de sentiments, qui présente encore des problèmes ouverts, comme nous le verrons par la suite dans ce manuscrit.

2.2 Traitement du texte pour l'analyse de sentiments

Avant de pouvoir réaliser de l'analyse de sentiments sur des tweets, il faut tout d'abord déterminer comment traiter et représenter le texte. Dans cette partie, nous verrons tout d'abord les caractéristiques des tweets, puis nous traiterons leurs méthodes de représentation.

2.2.1 Représentation des tweets

Afin de pouvoir utiliser des chaînes de caractères en apprentissage automatique, il faut au préalable pouvoir représenter le texte sous la forme d'un vecteur. Formellement, un texte peut être représenté par un vecteur $T = \{t_1...t_n\}$, de n *tokens*[2] t_i, appartenant

2. Un *token* est une instance d'une séquence de caractères dans un document, regroupés comme une unité sémantique utile pour le traitement Stanford NLP Group

tous à un dictionnaire. Ce dictionnaire est typiquement construit à partir d'un corpus $D = \{s_1...s_m\}$ de textes, qui sont soit les textes que l'on souhaite apprendre (e.g. les méthodes « sac de mots » de la section 2.2.1.1), soit les textes d'un corpus annexe, contenant souvent un plus grand volume de textes exemples (e.g. méthodes fondées sur les *embeddings* ou les modèles de langue en sections 2.2.1.3 et 2.3.3). Ce dictionnaire est composé de m *tokens* s non dédoublonnés. Le but de la représentation du texte est alors de trouver pour chaque *token* du dictionnaire une valeur numérique (score ou vecteur selon la méthode de représentation), pour le fournir ensuite en entrée d'un algorithme d'apprentissage.

Les méthodes de représentation des tweets, et plus généralement du texte peuvent être découpées en trois catégories : tout d'abord les représentations avec des « sacs de mots » puis les représentations fondées sur les *embeddings*, et enfin, les méthodes liées aux modèles Transformer, appelés modèle de langages. Dans cette partie, nous détaillons ces trois approches.

2.2.1.1 Sac de mots

Si depuis l'avènement des réseaux de neurones profonds, les méthodes de représentations du texte en TAL sont principalement fondées sur les *embeddings* et les modèles de langage, les méthodes « sac de mots » (BOW, pour *Bag-of-words* en anglais) sont longtemps restées les plus populaires dans la littérature. Celles-ci permettent d'associer un vecteur de réels à chaque document, ce qui facilite le traitement automatique du texte. Ici, à chaque *token* $s \in D$ est associé un score. Pour de l'analyse de sentiments sur Twitter, chaque document d correspond à un tweet faisant partie d'un jeu de données D. Ainsi, chacun des tweets d sera représenté par un vecteur T contenant les *tokens* présents dans D, associé à leur score t_i.

Plusieurs méthodes de calcul de score existent pour les « sacs de mots » :

Binaire (*One-hot encoding*) [111, 112]
: À chaque document correspond un vecteur $T = \{t_1, t_2, ..., t_n\}$, dans lequel chaque *token* $s \in D$ aura pour représentation t_i 1 si le *token* est contenu dans le document, et 0 s'il est absent.

Décompte d'occurrences
: Plutôt qu'une représentation binaire (présence ou non d'un *token*), il est possible d'utiliser le nombre d'occurrences de chaque *token*. Cela signifie que chaque *token* s est représenté par le nombre de fois où il apparaît dans le document $d \in D$.

Fréquence [111, 112]
: Afin de représenter un document, il est également possible d'associer à chaque *token* un score correspondant à la fréquence du terme $s \in D$ dans le document d. Cette fréquence correspond à $tf(d, t)$ calculé selon l'équation 2.1.

$$TF_t = \frac{Nombre\ d'occurrences\ de\ t\ dans\ d}{Nombre\ total\ de\ mots\ dans\ d} \tag{2.1}$$

TF.IDF [66]
: L'utilisation de la fréquence seule peut causer un biais. En effet, avec la fréquence, les termes très présents dans les documents auront un poids plus élevé

que ceux peu présents. Pourtant, un terme qui apparaît dans de nombreux documents n'est pas très discriminant et doit avoir moins de poids qu'un terme qui apparaît dans peu de documents. Afin de pallier ce problème, il est possible d'utiliser le produit TF.IDF (équation 2.3) [66]. L'utilisation du TF.IDF (pour *Term Frequency* et *Inverse Document Frequency*) a été très utilisé en Recherche d'informations [153]. Contrairement à l'utilisation de la fréquence seule, le TF.IDF permet de donner plus d'importance aux mots peu présents dans le corpus grâce à leur pondération par l'IDF (voir équation 2.2). En effet, le score IDF correspond au score de « rareté » d'un *token* s. En d'autres termes, il s'agit de l'importance d'un mot dans un document d, par rapport au corpus complet de documents D.

$$IDF_s = \log \frac{|D|}{|\{d \in D : s \in d\}|} \leftrightarrow IDF_s = \frac{Nombre\ de\ documents}{Nombre\ de\ documents\ avec\ s} \quad (2.2)$$

$$TF.IDF_{s,d,D} = TF_{t,d} \times IDF_{s,D} \quad (2.3)$$

Afin de minimiser la taille du dictionnaire, il peut être judicieux d'appliquer une étape de lemmatisation [3] ou de racinisation [4] au texte, ce qui permet de réduire le nombre d'entrées. Il est également fréquent de dresser une liste de *stop words*, mots nombreux dans les corpus mais « vides de sens », tel que les articles, ou les pronoms.

En plus des différentes alternatives de score, il est également possible d'utiliser des n-grammes plutôt que des mots pour composer le vecteur. Les n-grammes sont des séquences de n mots adjacents contenus dans un document d. Les représentations vectorielles sont alors composées de bigrammes (pour $n = 2$), trigrammes (pour $n = 3$), ou plus généralement de n-grammes et non de *tokens* (pour $n = 1$, il s'agit d'unigrammes).

Les « sacs de mots » ont été très utilisés avec les approches traditionnelles d'apprentissage automatique. En effet, (Pak and Paroubek, 2010) comparent les représentations binaires d'unigrammes, bigrammes et trigrammes pour leurs tweets [109]. Quant à (Pang et al., 2002), ils ont pu comparer la représentation binaire avec la fréquence d'apparition de leur différentes caractéristiques, en se servant des unigrammes, des bigrammes, d'une combinaison des deux, ou encore de caractéristiques linguistiques, comme la présence d'adjectifs [111]. L'utilisation de la fréquence pour le TAL était très répandue avant l'apparition des *embeddings*. En effet, plusieurs études de la littérature ont employé cette méthode de représentation du texte [7, 75, 86, 88, 91, 111]. (Barbosa and Feng, 2010) ont fait de la détection de sentiments dans des tweets en utilisant la fréquence d'apparition de deux catégories de caractéristiques : des *meta-caractéristiques*, comme les *POS tags* [5] ou la subjectivité, et des caractéristiques liées à la syntaxe des

3. Retrait de la flexion des mots
4. Transformation d'un mot en sa racine
5. *Part-of-speech*, il s'agit de la nature d'un mot d'un point de vue grammatical (nom, verbe, adjectif, etc.

tweets, comme les hashtags[6], les URL, ou encore les retweets[7] [7]. (Lambert et al., 2016) comparent cette fois l'utilisation du TF.IDF avec une méthode de boosting, à l'utilisation d'un *embedding* avec un CNN [75] (décrit en sous-section 2.3.2).

Ces types de représentations ont l'avantage d'être rapides à calculer, puisqu'ils ne nécessitent pas d'apprentissage au préalable. Néanmoins, il s'agit de méthodes coûteuses en mémoire, car chaque document est associé à un grand vecteur creux de la taille du vocabulaire ou du nombre de n-grammes. En effet, chaque document d, ou tweet, ne contient qu'une petite partie du vocabulaire présent dans le corpus D. De plus, les représentations des *tokens* t ne correspondent qu'à leurs utilisations dans D. L'utilisation d'*embedding* permet de corriger ces problèmes.

2.2.1.2 *Embeddings*

Nous venons de voir que les méthodes « sac de mots » permettent de représenter le texte en associant un vecteur T à chaque document d, dans lequel chacune des dimensions correspond à un mot du corpus D. Bien que ce type de représentation possède des avantages, les vecteurs des documents se trouvent être des vecteurs creux. Les méthodes d'*embedding* quant à elles, associent un vecteur de dimension n fixe pour chaque *token* représenté. Ces vecteurs associés aux *tokens* sont préalablement appris sur de grandes quantités de données annexes. Les algorithmes utilisés pour cet apprentissage permettent de saisir la sémantique des mots. Deux mots proches sémantiquement seront donc proches dans l'espace de représentation.

Les travaux sur ces types de représentations vectorielles, dits *embedding*, ont débuté en 2003 avec (Bengio et al., 2003) [1]. C'est en 2013 que (Mikolov et al., 2013) publient une première version du Word2vec, un *embedding* de mots en 300 dimensions fondé sur les *Continious Bag-of-words*[8] (CBOW). Les schémas en figure 2.1 montrent le fonctionnement du CBOW, ainsi que du skip-gram, autre algorithme d'apprentissage du Word2vec décrit ci-dessous. Nous pouvons voir que grâce aux *tokens* précédents (w_{t-2} et w_{t-1} sur le schéma) et suivants (w_{t+1} et w_{t+2} sur le schéma), le *token* w_t est prédit.

Puis, la même année (Mikolov et al., 2013) présentent la version finale du Word2vec [99], fondée cette fois sur les *skip-gram*[9]. Encore une fois, nous pouvons voir sur la figure 2.1 qu'à partir du *token* w_t, le contexte w_{t-2} et w_{t-1} est prédit. Suite au Word2vec, et aux algorithmes CBOW et skip-gram, d'autres méthodes de *word-embeddings* ont été publiées, nous pouvons notamment citer GloVe [115] (pour *Global Vector*) de Stanford, ou encore FastText [16] de Facebook.

Les *embeddings* permettent d'améliorer la représentation du texte en TAL puisqu'ils permettent de capturer la sémantique des mots. Ils sont donc très utilisés avec les réseaux de neurones. Par exemple, (Kim, 2014) a représenté le texte grâce à l'utilisa-

6. Mot-dièse en français, ce sont des mots précédés du symbole « # » qui correspondent à une thématique attribuée au message
7. Message ayant été reposté. Les retweets sont précédés du tag « RT »
8. Prédiction d'un mot selon son contexte
9. Prédiction du contexte selon un mot en entrée

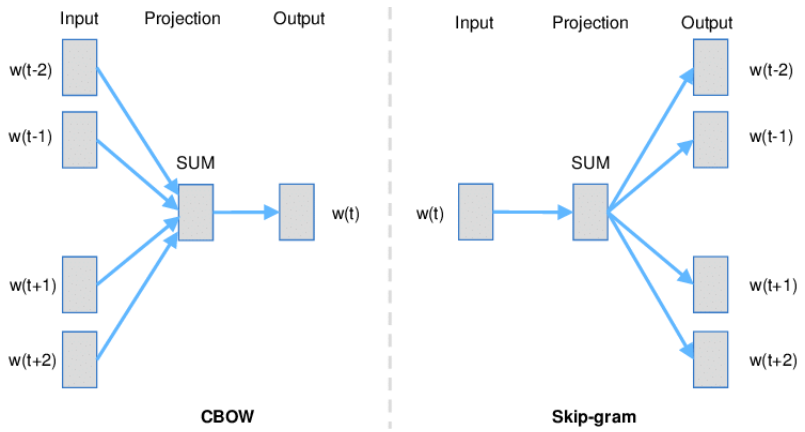

FIGURE 2.1 – Fonctionnement du *Continious Bag-of-words* et du *skip-gram*, extrait de [77]

tion du Word2vec, utilisé avant l'application de convolution (décrite dans la section suivante) [73]. (Lambert et al., 2016) ont utilisé un modèle pré-entraîné de GloVe, encore une fois avec de la convolution [75]. Word2vec, GloVe et FastText peuvent tous les trois être utilisés avec une combinaison de réseaux de neurones [26]. Ces trois méthodes de représentation sont également utilisées par (Rosá et al., 2017) : en effet, différentes approches ont été comparées en utilisant l'algorithme du Word2vec, ainsi que les représentations pré-entraînées de GloVe et FastText [126].

Le principe des *embeddings* n'a pas été employé uniquement au niveau du mot. (Zhang et al., 2015) et (Zhang and LeCun, 2015) ont mis en place une représentation du texte au niveau du caractère [174, 173]. BidAF [134] se fonde également sur un *character embedding* combiné à deux autres types de représentation : un *word-embedding* et un *contextual embedding*.

Si les représentations en *embedding* sont principalement utilisées par des réseaux de neurones, il est également possible de les adapter pour une utilisation par des algorithmes d'apprentissage automatique traditionnels décrits dans la section suivante. (Yin and Jin, 2015) ont utilisé un *Word2vec* avec un SVM pour réaliser de la classification de sentiments. Chaque *token* étant représenté par un vecteur de taille n, chacun de ces vecteurs a dû être adapté afin qu'il puisse être pris en entrée par un SVM. Pour chaque *token* d'un document, chacune des dimensions de son vecteur a, dans un premier temps, été additionné. De cette manière, les *tokens* sont représentés par une valeur unique, et non plus par un vecteur de taille n. Ainsi, la représentation obtenue est similaire aux « sacs de mots » (un score par *token*). (Lilleberg et al., 2015) ont également utilisé *Word2vec* avec un SVM, mais cette fois en combinant les vecteurs de mots avec le score $TF.IDF$ du même mot [84].

Les *embeddings* permettent d'obtenir une représentation vectorielle pour chaque *token* ou mot, en revanche ils ne permettent pas de gérer les homographes [10], les *tokens* faisant partie de mots ayant des sens complètement différents, ou les utilisations d'un même mot dans des contextes différents. Par exemple, « est » (flexion du verbe être) et « est »

10. Mots ayant la même orthographe, mais des sens différents

(point cardinal) seront représentés par le même vecteur, et ce quel que soit le contexte dans lequel ils apparaissent, alors qu'ils ont des sens biens différents. Avec l'apparition des modèles Transformer aujourd'hui à l'état de l'art, l'utilisation des embeddings tels que nous venons de les voir s'est raréfié, au profit de modèles de langages.

2.2.1.3 Modèle de langage

Les modèles de langage (ou modèle de langue) font à l'origine référence aux distributions de probabilités sur des *tokens* ou des séquences de *tokens* dans une langue. Cependant, actuellement et depuis l'apparition du bloc Transformer [151], lorsque les modèles de langage sont mentionnés, il s'agit généralement d'une référence à ce type de modèle.

Les modèles Transformer sont actuellement les plus performants pour la représentation du texte puisqu'ils permettent de capturer des relations sémantiques et syntaxiques. Ceci est notamment permis par l'utilisation de l'auto-attention, c'est-à-dire la mesure des relations entre les différents *tokens* d'une même séquence. Cependant, si les modèles Transformer constituent une avancée majeure, c'est grâce à l'utilisation d'un même modèle à la fois pour l'apprentissage de représentation du texte, et pour la tâche de classification ou de régression en aval.

De ce fait, même si ces méthodes visent à fournir une représentation sémantique riche et universelle d'une langue, leur utilisation est très dépendante de la tâche de TAL réalisée. Pour cette raison, nous choisissons de détailler ces méthodes en section 2.3.3.

Cependant, quel que soit le type de représentation sélectionnée, il doit être adapté au traitement des tweets. En effet, les tweets contiennent un vocabulaire particulier, tel que des abréviations, des hashtags, des mentions d'utilisateurs, etc... Dans la partie suivante, nous verrons ce vocabulaire plus en détail, ainsi que les différentes manières de le traiter.

2.2.2 Caractéristiques et traitement des tweets

Un tweet avait une longueur maximum de 140 caractères jusqu'en 2017, et est maintenant passé à 280 caractères. Des millions d'utilisateurs publient chaque jour des messages sur Twitter via n'importe quel appareil (smartphone, tablette, ordinateur...) [75]. De nombreux sujets sont abordés dans les tweets, avec un langage propre aux réseaux sociaux. En effet, la plupart des messages publiés contiennent fautes d'orthographes et abréviations, ceci étant dû à la limitation du nombre de caractères et à l'utilisation de smartphones, qui peuvent gêner la saisie de messages et causer des fautes de frappe. De plus, les messages sont écrits dans un registre familier [46, 75]. Il est donc nécessaire d'appliquer un certain nombre de pré-traitements aux tweets, afin de supprimer le plus de bruit possible, tout en s'adaptant à leur mode de représentation.

À l'écrit, et particulièrement dans un tweet, l'utilisation de majuscules permet de mettre certains mots en relief. Cela peut passer par l'ajout de majuscules en milieu de phrase lorsque ce n'est pas nécessaire, ou encore par l'écriture des mots entièrement en lettres capitales. Il s'agit donc d'une information importante puisqu'elle met

en évidence l'intention du locuteur lors de sa rédaction d'un message sur Twitter. Pourtant, ne pas normaliser le texte conduirait à un nombre de caractéristiques extrêmement élevé, car les formes « EVERYWHERE », « Everywhere », « everywhere », et même « eVeRyWHeRe » correspondent à quatre *tokens* différents. (Rosá et al., 2017) et (Lambert et al., 2016) proposent donc de normaliser la casse, et de passer tout le texte en minuscule, réduisant ainsi le nombre de caractéristiques à prendre en compte [75, 126]. Cela permet de limiter le nombre de *tokens* contenus dans les vecteurs des représentations BOW ou de faciliter la correspondance pour les *embeddings*.

Dans certains cas, le message peut contenir des répétitions de caractères pour marquer l'emphase sur le message véhiculé. Ceci correspond à l'allongement de mot (tel que « Gooooooooooood ») ou la répétition de ponctuation (« !!!!!! »). Cette fois encore, un problème de représentation pour les « sacs de mots » et les *embeddings* se pose, puisque le nombre de caractères ajoutés dépendra de l'utilisateur. Cela multipliera donc les entrées dans le dictionnaire et rendra presque impossible la présence du *token* dans l'*embedding*. Pour simplifier le traitement de mots contenant des répétitions de caractères, il est possible de ne conserver que les deux premiers caractères pour un caractère répété trois fois ou plus [46, 75], ou encore de n'en conserver qu'un seul [126].

En cas de retweet, un tag est présent en début de tweet permettant d'indiquer que le message a été reposté. Cette information peut être exploitée en tant que caractéristique [7] ou supprimée [54].

Les tweets peuvent intégrer des mentions d'utilisateurs, des liens vers des pages web (URL), ou encore des hashtags (ou « mot-dièse »). En effet, Twitter permet de mentionner un utilisateur en précédant son nom d'un « @ », ce qui crée un lien vers le compte mentionné. Ces mentions peuvent être supprimées [70] ou encore remplacées par un tag [46, 75, 126]. Concernant les URL, elles ne sont pas utiles à l'analyse de sentiments, et vont au contraire gêner l'analyse du tweet pour cette tâche. Elles peuvent donc être supprimées directement [54, 70, 75, 126] ou remplacées par un tag [46], ce qui permet de garder une trace du tweet d'origine. Pour les hashtags, ce sont des mots précédés du symbole « # » qui correspondent à une thématique attribuée au message. Deux choix sont possibles concernant leur traitement. D'une part, ils peuvent être supprimés totalement et remplacés par un tag [70]. Ceci permet de faciliter la représentation informatique du texte. En effet, les hashtags contiennent souvent des concaténations de mots ou encore des abréviations. Il est compliqué d'apprendre une représentation vectorielle de ces derniers dans la mesure où chaque utilisateur utilisera une syntaxe qui lui est propre. En supprimant les hashtags, ils seront donc remplacés par un seul et même terme, ce qui facilitera l'obtention d'une représentation vectorielle. D'autre part, il est également possible de supprimer uniquement le caractère « # » et de conserver le reste du hashtag [75]. Cette méthode permet de conserver les informations du hashtag, qui peuvent être importantes pour l'analyse de sentiments. En effet, la durée de vie des hashtag étant assez courte, il se peut qu'ils contiennent des informations importantes. Enfin, dans le but d'accéder aux informations contenues dans les hashtags, ceux-ci peuvent être découpés afin d'être séparés en mots [96]. Nous reviendrons plus en détail sur ces méthodes de normalisation dans le Chapitre 3.

2.3 Approches pour l'analyse de sentiments

Une fois les tweets représentés, la tâche d'analyse de sentiments peut être effectuée. Pour réaliser de la classification de sentiments, trois grandes approches se distinguent : les approches traditionnelles d'apprentissage automatique, les réseaux de neurones classiques utilisés pour la classification du texte, et enfin les modèles profonds fondés sur le bloc Transformer [151] vu en sous-section 2.2.1.3. Dans cette partie, nous introduisons ces trois méthodes, en commençant par les approches traditionnelles. Ensuite, nous abordons les différents types de réseaux de neurones adaptés au traitement du texte, et nous terminons par détailler les modèles profonds utilisant la couche Transformer, et en particulier le modèle BERT.

2.3.1 Approches traditionnelles d'apprentissage automatique

Les approches traditionnelles d'apprentissage automatique en TAL permettent de réaliser de la classification de texte, notamment de la détection de sentiments. Pour ces algorithmes, le texte doit être représenté sous forme numérique, et donc par un vecteur calculé en amont. Dans cette partie, nous décrivons ces approches en nous focalisant sur les deux algorithmes les plus utilisés pour l'analyse de sentiments tout en voyant les différentes utilisations de ces algorithmes.

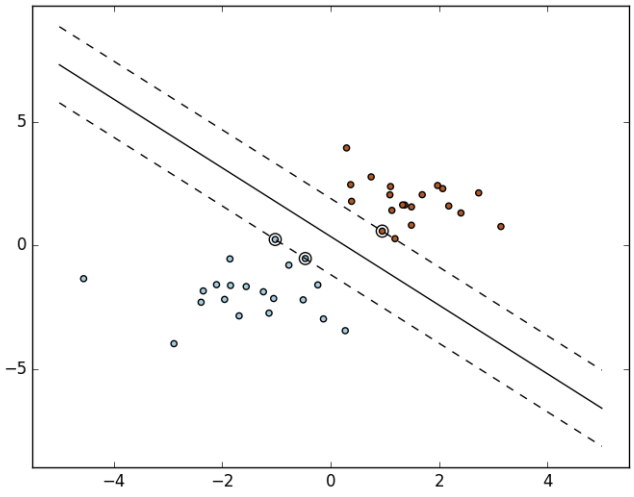

FIGURE 2.2 – Classification binaire avec un SVM, extrait de scikit-learn

Naive Bayes (NB) est un classifieur probabiliste fondé sur le théorème de Bayes et très employé en classification de texte. Il repose sur les probabilités conditionnelles. Utilisé dans le cadre de l'analyse de tweets, il permet de classifier les textes en fonction des mots qu'ils contiennent, et de leur probabilité d'appartenance à l'une des classes. NB a été très utilisé pour la classification de texte, et notamment en analyse de sentiments. (Wang and Manning, 2012) comparent l'utilisation de NB avec les SVM sur différents jeux de données [159]. Le *Support Vector Machine* (SVM) est un algorithme très utilisé pour réaliser de l'analyse de sentiments. Dans le cadre de l'analyse de tweets, l'objectif des SVMs est de trouver un hyperplan permettant de séparer les classes dans

l'espace de projection grâce au vocabulaire vectorisé des tweets, et ce, de la manière la plus optimale possible (voir figure 2.2 [11]). Ainsi, (Wang and Manning, 2012) ont démontré que NB obtenait de meilleures performances sur les textes courts, alors que sur les textes longs, les SVM étaient meilleurs. (Rosá et al., 2017) ont pu comparer sur des données annotées selon quatre polarités (positif, négatif, neutre, none) l'utilisation de SVM avec un réseau de neurones convolutif, ainsi qu'à une méthode hybride combinant les probabilités des classes du SVM et du CNN [126]. Bien que mauvais sur les tweets neutres, le modèle hybride a obtenu les meilleures performances globales, dépassant ainsi le SVM et le CNN utilisés seuls.

(Pang et al., 2002) ont comparé trois algorithmes : NB, les SVM et l'entropie maximale [111]. À ces trois algorithmes ont été associés des unigrammes (un seul mot), bigrammes (sous-séquence de deux mots), et d'autres familles de caractéristiques telles que le *POS tag* (nature grammatical d'un mot). Suite à ces comparaisons, il s'avère que pour ces trois algorithmes, les unigrammes offrent de meilleures performances, pouvant être améliorées en les combinant avec d'autres caractéristiques. Les performances des algorithmes sont également meilleures dans le cas d'utilisation de la représentation binaire, plutôt que la fréquence. À l'inverse, les résultats obtenus par (Dave et al., 2003) indiquent le contraire : l'utilisation de bigrammes et trigrammes offre de meilleures performances [30]. (Pak and Paroubek, 2010) ont voulu déterminer les meilleurs paramètres pour une utilisation sur des données Twitter. L'utilisation de bigrammes a cette fois obtenu les meilleures performances, ainsi que l'algorithme NB, en comparaison avec les SVM et les CRF [12] [109].

Les résultats obtenus avec ces différentes approches ne sont donc pas forcément stables, et dépendent des conditions (tâche et type de données) dans lesquelles elles sont appliquées. Cependant concernant l'analyse de sentiments sur des données issues de Twitter, c'est l'utilisation conjointe de bigrammes et de l'algorithme NB qui a obtenu les meilleurs résultats selon (Pak and Paroubek, 2010) [109]. Si les approches traditionnelles de l'apprentissage automatique ont permis au global d'obtenir de bonnes performances pour la classification de texte, elles ont été dépassées par les réseaux de neurones. Dans la partie suivante, nous voyons quelles sont les types de réseaux de neurones spécialisés dans le traitement du texte.

2.3.2 Réseaux de neurones récurrents et convolutifs

L'utilisation des réseaux de neurones et des modèles profonds a repoussé les limites des approches traditionnelles d'apprentissage automatique. Dans un premier temps, cela fut permis grâce au développement d'architectures spécialisées dans le traitement des séquences : les RNN (*Recurrent Neural Network*) [129], puis les LSTM (*Long-Short Term Memory*) [57] et les GRU (*Gated Recurrent Unit*) [22], qui sont tout deux inspirés des RNN. En effet, ces types de réseaux ont été conçus pour le traitement de séquences : grâce à leur mémoire et à leur principe de récurrence, ils sont adaptés au

11. https://scikit-learn.org/stable/modules/svm.html

12. *Conditional random fields*, Champs aléatoires conditionnels en français, il s'agit d'une classe de modèles statistiques qui sont utilisés en reconnaissance des formes et plus généralement en apprentissage statistique. Ils permettent de prendre en compte l'interaction de variables voisines et ils sont souvent utilisés pour des données séquentielles Datafrance

traitement du texte. Les CNN (*Convolutional neural network*) [79], modèles spécialisés dans le traitement des images, ont également été adaptés au traitement du texte. Tout comme les approches traditionnelles d'apprentissage automatique, les réseaux de neurones nécessitent que le texte soit représenté sous forme de vecteur. Dans cette partie, nous voyons de quelle manière les réseaux de neurones ont permis d'améliorer la classification de sentiments. Nous présentons tout d'abord chacun des types de réseaux, puis nous montrons leur utilisation pour le traitement du texte et l'analyse de sentiments.

2.3.2.1 Description des réseaux de neurones pour le texte

Les réseaux de neurones artificiels sont des systèmes inspirés des réseaux de neurones biologiques. Leur apprentissage est réalisé grâce à la rétropropagation du gradient. En effet, lorsqu'un réseau de neurones réalise une prédiction, celle-ci est comparée à la vérité terrain. L'erreur entre la sortie du réseau et la vérité terrain est ensuite calculée. Puis cette erreur est propagée vers l'arrière et les poids sont mis à jour. Cela signifie que les poids de chaque neurone sont mis à jour en partant de la dernière couche vers la première. Le but de cette rétropropagation est de converger vers un minimum global, qui correspond alors à une configuration optimale des poids. Dans cette partie, nous présentons les différents réseaux de neurones spécialisés dans le traitement du texte, et nous décrivons leur fonctionnement.

RNN

Les RNN (pour *Recurrent Neural Network*) [129] sont un type de réseaux dont les neurones sont inter-connectés de manière récurrente [171]. Ce principe de récurrence permet de simuler une « mémoire » grâce à l'état interne (*internal state*) tout au long de l'apprentissage. Ceci rend ce type de réseaux propice au traitement de données séquentielles, comme la vidéo, le son, ou encore le texte. En effet, la mémoire est telle que la même tâche est appliquée à chaque élément de la séquence, tout en prenant en compte les éléments vus précédemment. La figure 2.3 montre un exemple de RNN : à gauche, la représentation du réseau sous forme de cycle et à droite, une représentation dépliée avec trois étapes (pour une séquence de trois éléments). Ici, x_t correspond au vecteur d'entrée à l'étape t. L'état caché h_t (*hidden state*) est calculé à partir de l'état caché de l'étape précédente h_{t-1}, ainsi qu'à partir du vecteur d'entrée x_t. W^{hx} correspond à la matrice de poids utilisée pour conditionner le vecteur d'entrée x_t, et W^{hh} correspond quant à lui à la matrice de poids liée à l'état caché précédent h_{t-1}. Dans l'équation 2.6, la fonction d'activation f utilisée est généralement une fonction de tangente hyperbolique (équation 2.4) ou une fonction ReLU (équation 2.5). y_t correspond au vecteur de probabilité de la sortie sur le vocabulaire à l'étape t (équation 2.7). Enfin, W^{yh} correspond à la matrice de poids de la mémoire h_t à l'étape t. Dans la pratique, l'utilisation des RNN pour le traitement de séquences est limitée. En effet, dans le cas de dépendances de longue distance, c'est-à-dire éloignées dans la séquence, les RNN font face au problème de disparition du gradient (*vanishing gradient*) et d'explosion du gradient (*exploding gradient*), ce qui rend le traitement de ces dépendances plus complexe. En effet, plus la séquence est longue, plus le modèle aura d'étape et les dépendances seront éloignées. Ceci peut avoir pour conséquence la diminution très rapide du gradient, jusqu'à devenir très proche de 0 ou encore son

augmentation très rapide. Ce problème de gradient a donné lieu au développement des LSTM, une sous-classe de RNN adaptée aux dépendances de longue distance.

$$f\left(W^t x\right) = tanh\left(W^t x\right) = \frac{e^{W^{t_x}} - e^{-W^{t_x}}}{e^{W^{t_x}} + e^{-W^{t_x}}} \tag{2.4}$$

$$f\left(W^t x\right) = ReLU\left(W^t x\right) = max\left(0, W^{t_x}\right) \tag{2.5}$$

$$h_t = f\left(w^{hh} h_{t-1} + w^{hx} x_t\right) \tag{2.6}$$

$$y_t = softmax\left(w^{yh} h_t\right) \tag{2.7}$$

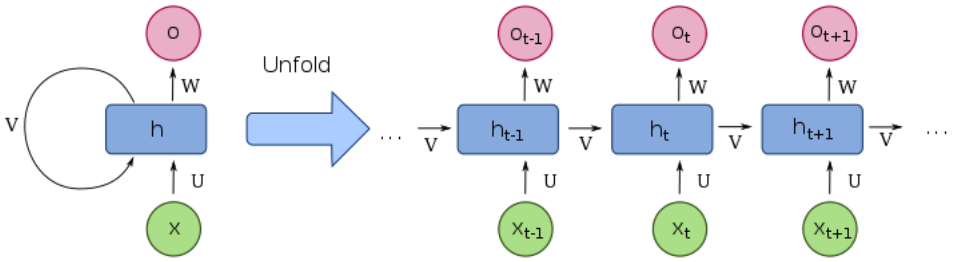

FIGURE 2.3 – Schéma d'un RNN, extrait de [171]

LSTM

Les LSTM (pour *Long-Short Term Memory*), présentés en figure 2.4 sont un type de réseaux faisant partie de la famille des RNN. Ils ont été conçus pour résoudre les problèmes de calculs liés au gradient initialement présents avec les RNN [57]. Leur structure est assez complexe. Tout d'abord, la mémoire est représentée par deux états : une cellule C (*cell state*) et un état caché h (*hidden state*). Le contenu de la mémoire est géré par trois portes qui ont des interactions entre elles :

Porte d'oubli : permet de décider quelles sont les informations à effacer de la mémoire

Porte d'entrée : permet de décider quelles sont les informations à mettre à jour dans C

Porte de sortie : sortie du réseau

La couche LSTM utilise à la fois les fonctions d'activation sigmoïde (équation 2.8) et la tangente hyperbolique (équation 2.4).

$$f\left(W^t x\right) = sigmoid\left(W^t x\right) = \frac{1}{1 + \exp\left(-W^t x\right)} \tag{2.8}$$

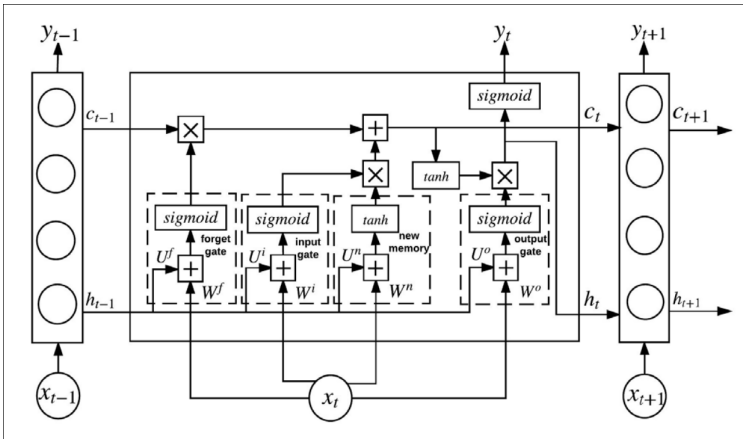

FIGURE 2.4 – Schéma d'un LSTM, extrait de [171]

GRU Les GRU (pour *Gated Recurrent Unit*) [22] sont un autre type de réseaux faisant parti de la famille des RNN. Il s'agit en fait d'une variation de LSTM plus simple que ces derniers car ils possèdent moins de composants. Comme nous pouvons le voir en figure 2.5, les GRU ne possèdent pas d'état de la cellule C (*cell state*) et n'ont que deux portes : une porte de reset et une porte de mise à jour. La porte de mise à jour est similaire à la porte d'entrée et à la porte d'oubli du LSTM. Quant à la porte de reset, elle permet de gérer la quantité d'information qui doit être oubliée.

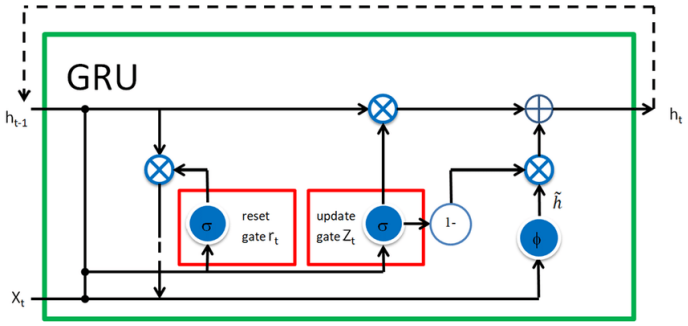

FIGURE 2.5 – Schéma d'un GRU, issu de [142]

CNN pour le texte

Initialement conçu pour l'analyse d'image, une architecture CNN (pour *Convolutional neural network*) [79] est constituée d'une ou plusieurs couches de convolution. Chacune de ces couches est généralement suivie d'une couche de *pooling* d'agrégation permettant de réduire la taille de la représentation, et donc le nombre de caractéristiques. (Kim, 2014) a mis en place de la convolution 1D (voir figure 2.6) afin de pouvoir effectuer plusieurs tâches de TAL, dont l'analyse de sentiments [73]. Pour cela, des filtres de convolution sont utilisés de la même manière que pour l'image, avec une fenêtre de taille h sur les *tokens* d'un texte. Ainsi, il s'agit de convolution 1D car il n'y a qu'une seule dimension à définir, celle de la taille de fenêtre h. La seconde dimension est en fait cachée et correspond au nombre de dimension de la représentation des *tokens* du texte.

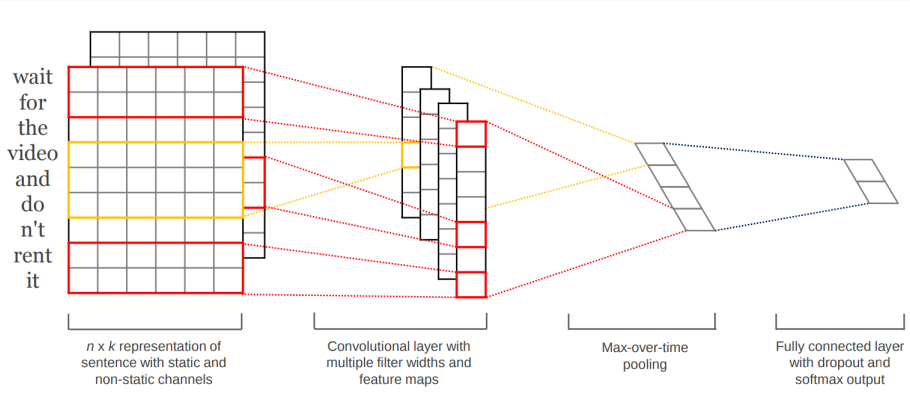

FIGURE 2.6 – Schéma d'un CNN pour le texte, extrait de (Kim, 2014) [73]

2.3.2.2 Utilisation des réseaux de neurones pour l'analyse de sentiments

Les réseaux de neurones spécialisés dans le traitement de séquences ont été utilisés dans le cadre de l'analyse de sentiments. (Ni and Cao, 2020) ont utilisé une combinaison de GRU et de LSTM pour la classification de sentiments [106]. Plus précisément, ils ont pu comparer les performances des RNN avec un LSTM, un GRU ainsi qu'avec la combinaison d'un LSTM avec un GRU, ce qui a permis de mettre en avant les défauts des RNN, corrigés par les GRU et LSTM. Pour capturer les relations sémantiques dans un texte, (Xu et al., 2016) proposent le CLSTM (*Cached Long-Short Term Memory*) une variation du LSTM qui, grâce à un système de cache, permet une division de la mémoire en plusieurs groupes [164]. Ces groupes sont gérés par différentes portes d'oubli, faisant office de filtres. Sur les jeux de données testés, le CLSTM obtient de meilleurs résultats qu'un LSTM classique ou bidirectionnel, ou même qu'un RNN.

Utilisés dans un premier temps par (Kim, 2014), les CNN pour le texte ont été appliqués par (Zhang et al., 2015) [174]. En effet, ils ont mis en place les ConvNets, c'est-à-dire de la convolution sur des données textuelles au niveau du caractère et non du mot. (Tang et al., 2015) ont proposé une architecture permettant l'apprentissage de représentations de documents, tout en prenant en compte les relations entre les phrases grâce à une couche de CNN ou de LSTM [145]. Cette couche permet donc d'apprendre les représentations des phrases (en utilisant des *embeddings*), puis les relations entre les différentes phrases sont ensuite apprises grâce à une couche de GRU.

(Wang et al., 2016) font suivre la convolution par une couche de LSTM , ce qui permet de capturer à la fois des informations locales, grâce au CNN, et à distance, grâce au LSTM, pour l'analyse de sentiments [158]. (Cliche, 2017) proposent également un modèle fondé sur une combinaison de CNN et de LSTM bidirectionnels pour réaliser de l'analyse de sentiments sur des données Twitter [26]. Après une phase de pré-traitement des tweets, les *word-embeddings* (voir partie 2.2.1.2) sont pré-appris en utilisant les algorithmes de *Word2vec* [99], *GloVe* [115], et *FastText* [16]. Une étape de supervision distante [46] permet ensuite de mettre les vecteurs à jour, et enfin, l'apprentissage supervisé peut débuter. (Park and Fung, 2017) réalisent de la détection de langage abusif sur Twitter en comparant trois structures [113] : la première avec un CNN fondé sur le mot, une deuxième avec un CNN au niveau du caractère, et enfin un HybridCNN

prenant à la fois les caractères et les mots. Deux approches sont comparées : la classification en une seule étape avec la solution proposée (HybridCNN, WordCNN ou CharCNN), et la classification en deux étapes en utilisant une combinaison de deux classifieurs binaires (la première étape consiste à classifier le langage abusif, puis la seconde à classifier les types de langage abusif). Ici, la classification en deux étapes permet d'améliorer les performances de modèles simples, et également de combiner différents types de classifieurs.

Bien qu'ils soient spécialisés dans le traitement des images, les CNN ont été très appliqués au texte, seuls ou combinés à d'autres types de réseaux. En effet, les CNN sont efficaces pour extraire les caractéristiques locales du textes [158], et sont plus rapides à entraîner que les LSTM [161]. En revanche, bien qu'ils soient populaires dans la littérature, les CNN ne semblent pas être les plus performants pour le traitement du texte [167]. En effet, (Yin et al., 2017) ont mené une étude comparant les CNN, ainsi que les différents types de RNN, montrant ainsi que les GRU obtenaient de meilleurs résultats [167]. Bien que les réseaux de neurones conçus pour le traitement des séquences ont permis d'améliorer la classification du sentiment, l'apparition des modèles Transformer a permis d'obtenir de bien meilleurs résultats. Dans la partie suivante, nous nous concentrons sur les modèles fondés sur la couche Transformer, qui ont une nouvelle fois bousculé les limites de l'analyse de sentiments.

2.3.3 Modèles Transformer

Les réseaux de neurones séquentiels, ainsi que les CNN pour le texte, ont été pendant quelques temps la meilleure alternative pour le TAL. Puis en 2017, (Vaswani et al., 2017) ont publié un article de référence, sur lequel s'appuient aujourd'hui les modèles à l'état de l'art : *Attention is all you need*. En effet, ce fut le début du bloc Transformer, et des modèles utilisant l'auto-attention [151].

Le Transformer est constitué d'une partie permettant l'encodage des données, composée d'encodeurs, et d'une seconde partie dédiée au décodage, composée de décodeurs. Chacun des encodeurs est composé de deux blocs. Tout d'abord, un bloc d'auto-attention, élément majeur du bloc Transformer, suivi d'un bloc de réseau à propagation avant (*feed-forward neural network*). Si le principe de l'attention consiste à mesurer le lien entre deux éléments de deux séquences, l'auto-attention correspond au même mécanisme appliqué à une seule séquence. En effet, l'auto-attention permet de déterminer les relations entre les différents *tokens* d'une même séquence. Ici, l'auto-attention est *multi-head*, cela signifie que les calculs sont effectués en parallèle par plusieurs têtes d'attention (figure 2.8). Pour les décodeurs, ils contiennent également un bloc d'auto-attention et un bloc de réseau à propagation avant, avec en plus un bloc d'attention Encoder-Decoder permettant de faire le lien entre la séquence encodée en entrée, et la séquence de sortie qui est décodée. Le schéma du Transformer est visible en figure 2.7.

Suite au développement du Transformer et surtout de l'auto-attention, deux premiers modèles de référence fondés sur cette architecture ont été publiés : OpenAI a publié GPT [118] (pour *Generative Pre-trained Transformer*), un modèle de langage auto-

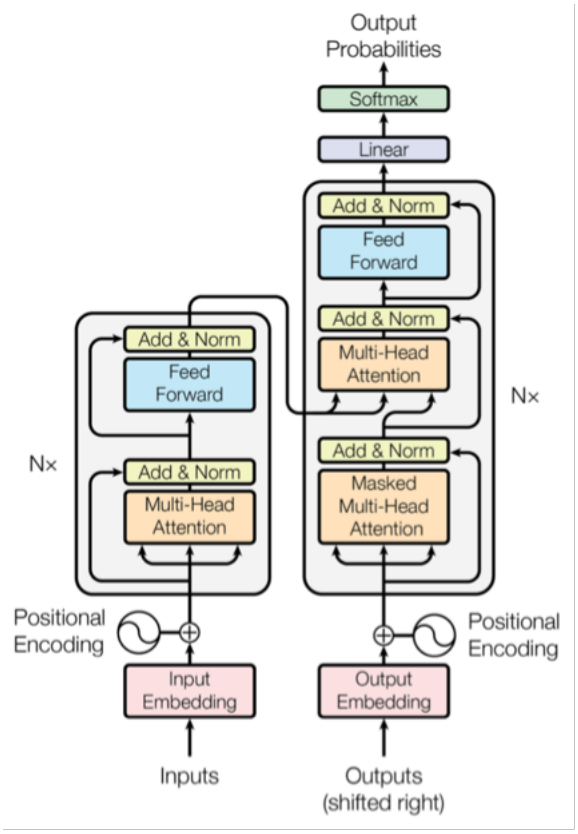

FIGURE 2.7 – Schéma du Transformer

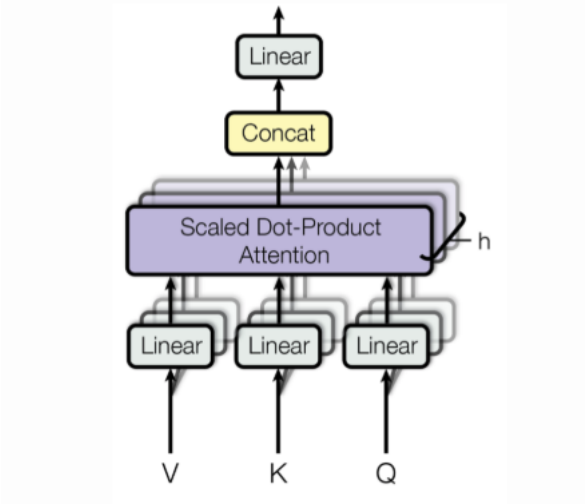

FIGURE 2.8 – Schéma de la *multi-head attention*

régressif, et Google a publié BERT [33] (pour *Bidirectional Encoder Representations from Transformers*), un modèle de langage bidirectionnel.

Les modèles auto-régressifs, comme GPT, sont composés uniquement de décodeurs. Ainsi, leur tâche d'entraînement est adaptée à leur structure : pour apprendre, le modèle tente donc de prédire un *token* en utilisant seulement les précédents. En ce qui concerne les modèles similaires à BERT, ils sont composés uniquement d'encodeurs

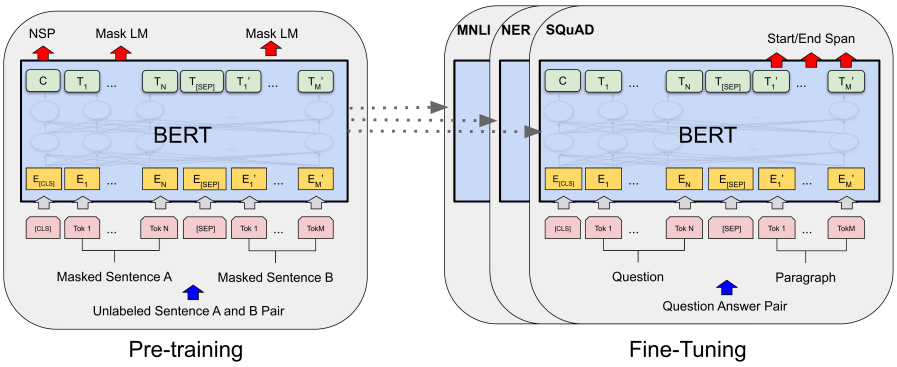

FIGURE 2.9 – Fonctionnement du modèle BERT, extrait de [33]

et sont entraînés en utilisant le MLM (*Masked language modeling*), qui consiste à masquer une partie du texte donné au modèle avec un *token* spécifique, et laisser le modèle trouver le *token* masqué en se basant sur le texte qui sert de contexte.

Avec ce type de modèle, l'utilisation d'une représentation du texte apprise au préalable n'est plus nécessaire, puisque l'architecture permet directement d'apprendre cette représentation de manière auto-supervisée. L'idée est donc d'entraîner les modèles Transformer sur de grandes quantités de données non annotées, afin d'obtenir une représentation de la langue. Les données utilisées sont généralement génériques, afin d'obtenir une représentation de la langue la plus générique possible. Ces modèles sont ensuite affinés sur les données cibles afin d'être adaptés à la tâche traitée.

BERT [33] a donc posé les bases des modèles Transformer bidirectionnels. Il s'agit d'un modèle composé de plusieurs couches de bloc Transformer (uniquement les encodeurs), publié par Google en 2019. À sa publication, le modèle est disponible en deux tailles : BERT-base et BERT-large. BERT-base est composé de 12 couches (de bloc Transformer), avec une sortie de taille 768, 12 têtes d'attention, et 110 millions de paramètres au total. Quant à BERT-large, il est composé de 24 couches, avec une taille de sortie de 1024, 16 têtes d'attention, et 340 millions de paramètres au total. Les deux modèles sont entraînés avec la même quantité de données. Cependant, avec plus de couches, le grand modèle contient plus de paramètres et est censé être plus efficace que le petit, bien qu'il soit plus long à entraîner.

BERT a été pré-entraîné grâce à l'auto-supervision. L'apprentissage auto-supervisé peut être considéré comme étant à la frontière entre l'apprentissage supervisé et non supervisé. Les tâches utilisées pour l'apprentissage du modèle sont deux tâches non supervisées : MLM et NSP. Le MLM, pour *Masked Language Model*, consiste à masquer aléatoirement un certain pourcentage des *tokens* d'entrée, avant d'essayer de les prédire. Le NSP, pour *Next Sentence Prediction*, consiste à prédire si une phrase B suit une phrase A (voir figure 2.9 pour le fonctionnement de BERT). Le BooksCorpus (800 millions de mots) [176] et le Wikipedia anglais (2 500 millions de mots) ont été utilisés pour le pré-entraînement de BERT.

Plusieurs analyses des couches de BERT et des têtes d'attention ont permis d'identifier que le modèle contenait des informations sémantiques [27, 63] et syntaxiques [24, 27,

48, 58, 63, 85, 155]. Il a également été démontré que certaines des têtes d'attentions pouvaient être retirées sans affecter les performances de l'attention. Cela conduirait à un changement de fonction des têtes d'attention restantes [155]. Enfin, l'analyse des couches de BERT a permis de révéler les informations contenues dans chacune d'entre-elles. Ainsi, les dernières couches contiennent des informations liées à la structure de la phrase, tandis que celles du milieu contiennent des informations syntaxiques, et les premières couches des caractéristiques syntaxiques [63, 85]. Cependant, si certaines études mettent en avant la structure du modèle, d'autres au contraire remettent au cause la forte corrélation entre les poids d'attention et les conclusions tirées [18, 62, 137, 162]. Ainsi, la littérature n'est pas encore claire sur ce point.

Après la publication de BERT et de GPT, de nombreux modèles ont suivi. BERT a donc été décliné en plusieurs versions. Par exemple, RoBERTa [89], est un modèle beaucoup plus gros que BERT. Il est entraîné sur dix fois plus de données que BERT, et possède 48 couches et cinq fois plus de paramètre. Ici, la tâche de NSP a été supprimée, car elle ne montrait pas d'intérêt pour l'entraînement du modèle. Pour le modèle AlBERT [76], la tâche de NSP a cette fois été remplacée par celle de SOP (*Sentence Order Prediction*), c'est-à-dire que le modèle doit déterminer l'ordre de paires de phrases. Il s'agit de variations de BERT, sans réelles modifications d'architecture.

Dans le cas d'OpenAI, le modèle GPT-2 [119] a également été publié. Pour ce modèle, la quantité de données d'entraînement a été grandement augmentée, ainsi que le nombre de paramètres. Puis plus récemment, GPT-3 [17] fut publié. Avec encore plus de données d'entraînement et plus de paramètres, ce modèle est le plus performant des trois modèles publiés par OpenAI.

D'autres modèles tels que le T5 [121], Transformer-XL [29], ELECTRA [25] ou encore DistilBERT [131] (cette liste est non exhaustive), tous fondés sur la couche Transformer, ont également été publiés. Le T5 (Text-to-Text Transfer Transformer) [121] utilise une approche texte-à-texte. Le modèle possède donc un décodeur bidirectionnel permettant de générer le texte en sortie. Le Transformer-XL [29] fonctionne sur un principe semblable à de la récurrence, la différence étant que les états cachés sont transmis plutôt que d'être réinitialisés à chaque segment. Pour ELECTRA [25], les masques du MLM sont remplacés par des *tokens* plausibles qui sont générés par un réseau partageant ses poids avec le modèle. Pour chaque *token*, le modèle doit donc déterminer s'il a été généré ou non. Enfin, dans le cas de DistilBERT, le modèle a été entraîné sur les prédictions de BERT et a donc appris à imiter ce modèle.

Comme nous venons de le voir, la plupart des modèles fondés sur les Transformer sont très proches du modèle BERT, puisqu'ils en sont dérivés. Par conséquent, nous nous concentrons principalement sur ce modèle. En effet, c'est BERT qui a été sélectionné comme modèle de référence pour réaliser nos expérimentations.

Même si les algorithmes classiques d'apprentissage automatique, utilisant ou non les réseaux de neurones, ont été beaucoup appliqués à l'analyse des sentiments, les modèles Transformer constituent l'état de l'art actuel. Dans (Li et al.) et (Hoang et al., 2019),

BERT a été utilisé pour l'analyse de sentiments fondée sur l'aspect [13]. En effet, (Li et al., 2019) l'ont utilisé comme représentation du texte, en ajoutant un classifieur en aval [82]. Plusieurs classifieurs ont été testés pour cette configuration : une couche linéaire, un GRU, de l'auto-attention ou encore un CRF. L'utilisation de BERT comme représentation a permis à chacun de ces classifieurs d'obtenir de meilleurs scores qu'en utilisant d'autres types de représentations. (Hoang et al., 2019) ont affiné BERT sur des données réelles et générées pour créer un classificateur de sentiments [56]. (Gao et al., 2019) ont également affiné BERT pour l'analyse de sentiments selon sur la cible [42]. Les modèles BERT, mais aussi RoBERTa, DistilBERT, T5 et XLNet ont été affinés et comparés pour l'analyse de sentiments sur les données IMDB [116]. Il a été montré que les modèles Transformer surpassent les LSTM bidirectionnels en précision.

2.4 Conclusion

Dans ce chapitre, nous avons vu que l'analyse de sentiments et la reconnaissance d'opinion consistait en deux tâches différentes, mais pourtant confondues dans la littérature. Bien que quatre parmi cinq informations du quintuplet nécessaire à une opinion soient considérées comme connu, en ne détectant que la polarité des tweets, nous nous rapprochons de la tâche d'analyse de sentiments. Afin de réaliser cette tâche d'analyse de sentiments, nous avons pu voir différentes approches : les approches traditionnelles d'apprentissage automatique, les réseaux de neurones pour le traitement de séquences, et les modèles fondés sur la couche Transformer.

Il est clair maintenant que les modèles Transformer sont l'état de l'art pour l'analyse de sentiments. Ils fonctionnent sur principe suivant : tout d'abord, un pré-apprentissage de représentation auto-supervisé est réalisé afin de représenter au mieux la sémantique d'un texte exprimé dans un langage donné. Puis la tâche de classification est effectuée grâce à l'utilisation d'une couche de décision en aval, qui peut être affinée de façon supervisée. Ceci permet de résoudre une tâche spécifique de classification par exemple, comme l'analyse de sentiments.

Or, comme nous l'avons expliqué dans ce chapitre, ces architectures sont complexes et surtout très gourmandes en ressources (données, mémoire et calculs). Ainsi, réaliser l'apprentissage de zéro est presque impossible lorsque l'on n'est pas une grande entreprise comme Google ou OpenAI, car ces ressources sont difficilement accessibles.

Cependant, réaliser cet apprentissage n'est pas forcément nécessaire. En effet, le pré-entraînement permet d'apprendre la représentation d'une langue et non d'un corpus de texte. Ainsi, les modèles mis à disposition librement (sur HuggingFace par exemple) pourraient suffire à résoudre un problème de TAL, grâce à l'utilisation du *fine-tuning*.

Dans le cas de la classification de sentiments dans des tweets, une question reste ouverte vis-à-vis de ce procédé. En effet, les tweets étant composés d'un vocabulaire spécifique (langage familier, abréviation, fautes, hashtag, etc), nous pouvons nous in-

13. Consiste à trouver à la fois la polarité de l'opinion et l'objet

terroger sur la pertinence des modèles de langues anglais pour traiter le vocabulaire de Twitter.

Nous proposons d'approfondir cette question dans le prochain chapitre, en nous intéressant à l'analyse de sentiments dans les tweets avec des modèles Transformer dans le cas où l'apprentissage d'une représentation spécifique aux tweets est impossible. Nous nous intéressons principalement à BERT [33], l'un des deux premiers modèles Transformer [151] publié. En effet, ce modèle a inspiré les nombreux modèles Transformer auto-encodeurs suivants.

Chapitre 3

Analyse de sentiment avec des ressources limitées

Résumé

Ce chapitre est consacré à la définition d'une ligne de conduite à suivre afin de faire de l'analyse de sentiments sur des tweets en ayant à disposition un unique GPU. Notre objectif est de mettre en place une série de lignes directrices maximisant les performances de BERT, tout en évitant le problème de l'oubli catastrophique grâce au gel des couches et au maintien d'un taux d'apprentissage faible. Plus précisément, il s'agit d'étudier différents types d'apprentissages (le *fine-tuning* et la poursuite du pré-apprentissage de modèles génériques, ainsi que l'utilisation d'un modèle uniquement entraîné sur des données Twitter), ainsi que le traitement d'éléments du texte propres à Twitter tels que les emojis, les URL, ou les mentions utilisateurs. Nous montrons que si poursuivre le pré-apprentissage d'un modèle générique sur des tweets permet d'atteindre des performances similaires à celles des modèles entraînés sur des données Twitter, le coût en ressources et en temps reste élevé. Il semble donc préférable d'affiner un modèle déjà appris sur des tweets lorsque cela est possible, dans notre cas BERTweet.

Sommaire

Analyse de sentiment avec des ressources limitées

Dans le chapitre précédent, nous avons vu que les réseaux de neurones profonds, et en particulier les modèles Transformer, sont actuellement à l'état de l'art pour l'analyse de sentiments. Les modèles les plus récents, fondés sur la couche Transformer [151], ont révolutionné le traitement du automatique du langage naturel (TAL) : GPT [118] et BERT [33] ont été les deux premiers modèles publiés, suivis par une série d'autres [17, 25, 29, 76, 89, 104, 119, 121, 130]. Ils ont permis d'améliorer les approches classiques d'apprentissage profond pour la tâche d'analyse de sentiments [116]. Nous avons donc sélectionné les modèles Transformer, et en particulier le modèle BERT, pour la poursuite de nos expérimentations.

Les modèles d'apprentissage profond ont besoin de beaucoup de ressources. Cela est particulièrement vrai pour les modèles Transformer qui ont besoin de données, de temps et de GPU pour être entraînés. Par exemple, l'un des modèles les plus populaires actuellement, GPT-3, est 1 000 fois plus grand que GPT-2 (175 milliards de paramètres pour GPT-3) et a été entraîné en utilisant l'équivalent de 355 GPU/année, ce qui représente 4,6 millions de dollars [1]. En effet, un tel apprentissage nécessiterait 355 ans en utilisant un Tesla V100, le GPU le plus rapide du marché lors de la publication de GPT-3. Malheureusement, peu de personnes peuvent se permettre ce genre d'entraînement et la majorité ne peut avoir accès qu'à un seul GPU pour entraîner des modèles.

Comme tout le monde ne peut pas avoir accès à cette quantité de CPU ou de GPU, il faut trouver une solution alternative. DistilBERT [130], une version distillée de BERT, donc plus petite et moins gourmande en ressources, et qui a appris à imiter BERT, pourrait être une solution au manque de ressources car il ne possède que 66 millions de paramètres. Mais ce modèle réalise également des performances inférieures à celles de BERT [116]. L'une des meilleures solutions est d'utiliser un modèle pré-entraîné, et de l'affiner sur les jeux de données cible pour le spécialiser [59]. Affiner un modèle pré-entraîné sur des données sources consiste à adapter ses poids en utilisant un jeu de données cible grâce à de l'entraînement supplémentaire. Cette méthode a été exploitée pour l'analyse des sentiments avec succès avec des modèles d'apprentissage profond et de Transformer [59, 64, 146, 165].

Le *fine-tuning* a également une faiblesse : il est sensible à l'oubli catastrophique [97]. En effet, lorsque les modèles d'apprentissage profond sont affinés, ils ont tendance à oublier les connaissances qu'ils ont acquises lors de leur pré-entraînement. Ceci est particulièrement vrai dans le cas des modèles Transformer tels que BERT, qui peuvent avoir des difficultés à s'adapter aux jeux de données cible [144].

Dans ce chapitre, nous proposons donc un ensemble de bonnes pratiques pour effectuer de l'analyse de sentiments sur Twitter, en utilisant des modèles Transformer à l'état de l'art lorsque les ressources sont limitées. La suite de ce chapitre est organisée

1. https://lambdalabs.com/blog/demystifying-gpt-3/

comme suit : nous présentons tout d'abord un panorama sur l'utilisation des modèles Transformer avec des ressources limitées. Puis, nous décrivons les jeux de données utilisés ainsi que le dispositif expérimental, avant de présenter nos recommandations, qui comprennent le choix de la taille du modèle et de l'approche d'apprentissage. Des solutions à l'oubli catastrophique sont ensuite proposées, et nous terminons par la manière de gérer les particularités du vocabulaire de Twitter.

3.1 Utilisation des modèles Transformer avec des ressources limitées

Au vu de l'impressionnante puissance de calcul nécessaire pour l'apprentissage d'un modèle Transformer, peu de personnes peuvent se permettre d'apprendre ce type de modèle depuis zéro. Il faut donc trouver des solutions permettant tout de même de réaliser de la classification en étant à l'état de l'art, malgré des modèles gourmands en données d'entraînement, temps d'apprentissage ou encore mémoire, dû au nombre de paramètres des modèles. Dans cette section, nous verrons qu'il est possible de réaliser notre tâche en utilisant tout de même ces modèles, en ayant des ressources limitées. Nous parlons tout d'abord de l'utilisation du *fine-tuning* (ou affinage du modèle), puis nous voyons la poursuite du pré-apprentissage du modèle, et enfin, nous terminons l'utilisation de modèles spécialisés sur notre type de données, lorsque ces modèles existent.

3.1.1 Utilisation du *fine-tuning*

Le *fine-tuning* peut être utilisé pour adapter un modèle générique. Celui-ci permet d'utiliser les connaissances acquises lors du pré-entraînement sur une tâche source, et de les appliquer sur une tâche cible. Concernant les modèles Transformer, qui sont gourmands et coûteux à entraîner, ils sont conçus pour être affinés, afin d'être adaptés aux jeux de données ainsi qu'aux tâches cibles.

Cependant, ce genre de méthode est souvent confronté à l'oubli catastrophique (*catastrophic forgetting*) : BERT oublie progressivement une partie des connaissances acquises pendant son pré-apprentissage. Concrètement, cela signifie que pendant l'affinage du modèle, la mise à jour des poids se fait brutalement, ce qui entraîne une stagnation des performances du modèle, ou encore un sur-apprentissage sur le jeu de données. Afin d'éviter l'oubli catastrophique, plusieurs solutions sont possibles. La première solution consiste à geler certaines couches du modèle, c'est-à-dire empêcher la mise à jour des poids de ces couches. Dans ULMFiT [59], (Howard and Ruder, 2018) recommandent d'utiliser le dégel progressif (*gradual unfreezing*), c'est-à-dire de commencer le *fine-tuning* en gelant toutes les couches sauf la dernière, puis de les dégeler progressivement, une par une, après chaque epoch. Le modèle est ainsi affiné jusqu'à convergence. (Nguyen et al., 2020) proposent le gel critique (*critical freezing*), une solution plus simple avec laquelle seules les premières couches du modèle sont gelées [105]. Plus précisément, (Lee et al., 2019) ont effectué plusieurs tests concernant le gel des couches du modèle, et ont obtenu les meilleurs résultats en affinant seulement le dernier quart du modèle [80]. Ainsi, les premières couches sont gelées, jusqu'à la neu-

vième couche pour BERT-base, et jusqu'à la dix-huitième couche pour BERT-large. Cette méthode force le modèle à conserver la majorité de ses connaissances, et à ne mettre à jour que ses dernières couches. Une autre solution pour limiter l'oubli catastrophique est d'utiliser un faible taux d'apprentissage. (Howard and Ruder, 2018) proposent ici d'utiliser le *slanted triangular learning rate* [59]. Cette méthode consiste à débuter l'apprentissage avec un taux d'apprentissage faible, puis à l'augmenter rapidement, avant de le diminuer à nouveau, lentement, comme illustré en figure 3.1. Selon (Nguyen et al., 2020), simplement conserver un taux d'apprentissage faible et constant permet d'éviter l'oubli catastrophique [105]. Dans ce chapitre, nous nous focalisons sur les solutions apportées par (Lee et al., 2019), c'est-à-dire geler toutes les couches sauf le dernier quart [80], ainsi que celles de (Nguyen et al., 2020) concernant l'utilisation d'un taux d'apprentissage faible [105]. Ces solutions sont proches de celles apportées par ULMFiT [59], qui concernent à la fois le gel des couches et le taux d'apprentissage. Elles sont cependant plus simples et rapides à mettre en place, puisque qu'aucun changement n'est effectué durant le *fine-tuning* du modèle.

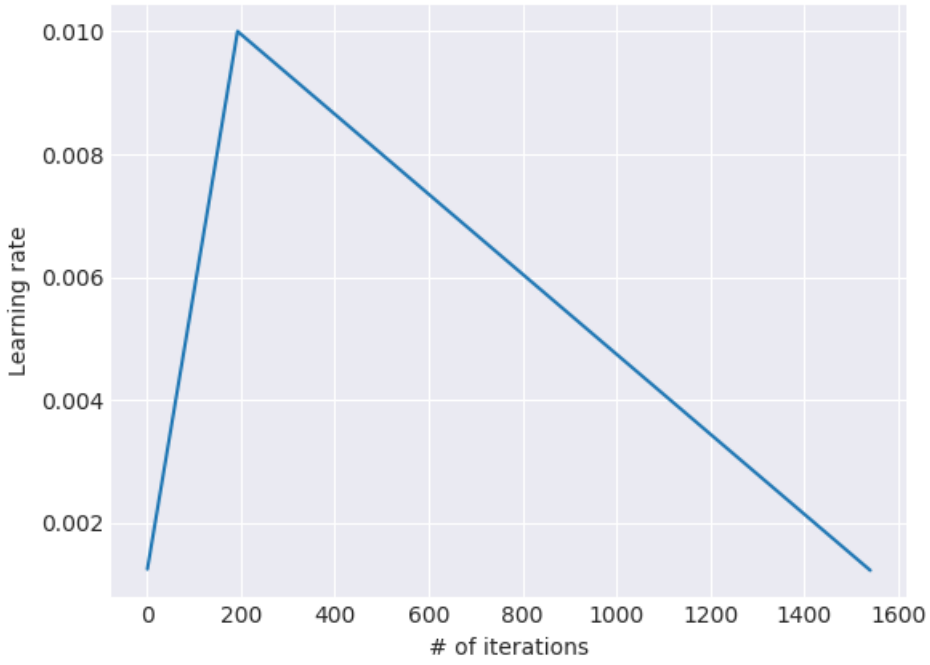

FIGURE 3.1 – *Slanted triangular learning rate*

Après avoir mis en avant les avantages du *fine-tuning*, ainsi que ses inconvénients, dans la partie suivante nous décrivons la poursuite du pré-apprentissage du modèle.

3.1.2 Poursuivre le pré-apprentissage du modèle

Les modèles pré-entraînés sur des données génériques correspondent à une représentation de la langue générique, commune, c'est-à-dire non spécialisée sur un thème ou un type de données en particulier. Ainsi, une autre solution possible, lorsque le manque de ressources ne permet pas l'apprentissage d'un modèle de zéro est de pour-

suivre l'entraînement d'un modèle générique sur le type de données cible [20, 144]. En faisant cela, la représentation générique de la langue se spécialise sur les spécificités des données cibles, dans notre cas sur les tweets.

Poursuivre le pré-entraînement d'un modèle consiste donc à continuer les tâches de pré-apprentissage sur des données supplémentaires. Par exemple, pour BERT, les tâches de MLM (*Masked language modeling*) et de NSP (*Next Sentence Prediction*) seraient utilisées. Puisque le modèle générique pré-entraîné subit un véritable second apprentissage sur un type de données spécifique, comme des tweets, cette méthode permet de l'adapter à ce type de données de manière plus efficace qu'avec du *fine-tuning*, où seul la tâche est apprise.

3.1.3 Utilisation d'un modèle spécifique quand cela est possible

Les deux solutions décrites précédemment s'appuient sur l'utilisation de modèles pré-entraînés, nous n'avons donc pas réalisé d'apprentissage de modèle de zéro. En effet, effectuer un tel apprentissage d'un modèle Twitter de zéro est presque impossible lorsque les ressources sont limitées. La publication de BERTweet [104] résout en partie ce problème, puisqu'il s'agit d'un modèle non pas générique, mais pré-entraîné sur des données Twitter. Ce modèle utilise la même architecture que BERT-base. Il est entraîné sur 850 millions de tweets (80GB) en utilisant la démojisation, qui consiste à remplacer un emoji par son texte. (Nguyen et al., 2020) se sont servis de 8 GPU V100 de 32GO pour entraîner le modèle, pendant 4 semaines, en utilisant un batch de 7 000 tweets pendant 40 epochs [104]. L'implémentation de RoBERTa [89] de la librairie fairseq [108] a été utilisée pour l'apprentissage du modèle.

Un modèle spécialisé sur les données issues de Twitter sera plus adapté aux tweets, et nécessitera seulement une étape de *fine-tuning* pour apprendre la tâche de classification. Si ce type de modèle existe pour BERT et les données Twitter, les modèles spécialisés restent rares. De plus, pour les modèles réutilisant les architectures des modèles Transformer existants, il y a un délai entre la publication du modèle générique et sa version spécialisé. En effet, le modèle BERT a été publié en 2019, et BERTweet seulement un an après, en 2020. Ainsi, apprendre les bonnes méthodes pour le *fine-tuning* de modèles génériques sur des données Twitter est également important, puisque les modèles Transformer spécifiques ne sont pas forcément disponibles, ou ne sont pas entraînés immédiatement après leur équivalent générique.

3.1.4 Discussion

Nous venons de décrire trois approches différentes permettant de gérer le manque de ressources pour une tâche de classification : l'affinage d'un modèle Transformer générique (BERT), la poursuite du pré-apprentissage du même modèle, et enfin, le *fine-tuning* d'un modèle Twitter. Intuitivement, nous pouvons considérer que la meilleure méthode est l'utilisation d'un modèle spécifique, entraîné sur le même type de données que les données cibles, comme BERTweet pour les données Twitter. En effet, l'utilisation d'un modèle spécialisé permet d'obtenir non seulement des poids plus proches

des données Twitter dans le modèle de langage, mais également des *tokens*[2] qui correspondent mieux au vocabulaire des tweets, avec ses caractères spéciaux, etc.

Si pour BERT, le modèle que nous avons sélectionné, une version Twitter est disponible, ce n'est pas le cas pour tous les autres modèles Transformer publiés. En effet, les modèles génériques sont plus courants que les modèles spécialisés. De plus, lors de la publication d'un nouveau modèle, seul la version générique sera disponible dans un premier temps. Il est donc nécessaire de pouvoir adapter ce type de modèle à des données Twitter, ou de manière générale, à des données non génériques.

Dans la suite de ce chapitre, nous proposons donc de comparer ces trois approches, de manière à mesurer l'impact de l'utilisation d'un modèle spécialisé. Nous avons testé ces approches en utilisant six jeux de données publiques, décrit ci-dessous. Dans la partie suivante, nous donnons des recommandations permettant de faire de l'analyse de sentiments en ressources limitées.

3.2 Protocole

Dans le but d'établir nos recommandations pour l'analyse de sentiments Twitter avec des ressources limitées, nous avons mis en place un protocole pour nos expérimentations. Ce protocole est défini ci-dessous.

3.2.1 Expérimentations

Puisque notre objectif est d'étudier les conditions expérimentales dans lesquelles on dispose de ressources limitées, toutes les expériences présentées ci-dessous ont été menées sur un seul GPU abordable et non destiné à être utilisé dans des centres de données : un NVIDIA GeForce GTX 1080 avec 8 Go de RAM.

Deux tailles de batch[3] différentes ont été utilisées, en fonction de la taille du modèle. Pour BERT-base, nous avons utilisé une taille de batch de 16, avec une accumulation de gradients de 2 (c'est-à-dire que les gradients de 2 batchs sont accumulés avant d'être retropropagés), pour obtenir une taille de batch finale de 32, qui est la plus grande taille de batch recommandée pour BERT [94] compatible avec notre GPU. Pour BERT-large, la taille de batch maximale que nous pouvions utiliser sans saturer le GPU était de 4. Nous avions donc une accumulation de gradients de 8, pour obtenir également une taille de batch finale de 32. Enfin, la classification est réalisée en utilisant uniquement le modèle BERT affiné.

Pour affiner nos modèles, nous avons dû établir un protocole expérimental afin de rendre les différents résultats comparables et d'éviter de tirer des conclusions faussées. Tout d'abord, toutes les expériences ont été menées en utilisant 10 découpages aléatoires stratifiés[4] de chaque base de données avec 80% pour l'entraînement et 20%

2. Instance d'une séquence de caractères dans un document, regroupés comme une unité sémantique utile pour le traitement Stanford NLP Group
3. Regroupement des données par lots
4. Les proportions de classes sont conservées dans chaque sous-ensemble.

pour le test. Les modèles ont été affinés sur chaque découpage pour un maximum de 20 epochs, jusqu'à atteindre la convergence de la fonction de perte d'apprentissage. Avec tous ces résultats, nous avons calculé un score moyen pour chaque modèle, ainsi qu'un écart-type. Afin d'évaluer les différentes méthodes, nous avons utilisé le score f1 [5], qui est la mesure la plus utilisée en analyse de sentiments. En effet, les jeux de données d'analyse de sentiments sont souvent déséquilibrés, et le score f1 permet de mieux gérer ce déséquilibre de classe. Dans la partie suivante, nous verrons les différents jeux de données que nous avons utilisés pour nos expérimentations.

3.2.2 Jeux de données pour l'analyse de sentiments

Pour nos expérimentations, nous avons dû nous constituer un ensemble de corpus. Nous nous sommes intéressés aux jeux de données publiques, d'une part parce qu'ils sont gratuits et facilement accessibles, mais également afin de pouvoir nous comparer à la communauté. Chacun de ces jeux de données est sélectionné pour leur tâche de classification. En effet, puisque nous faisons de l'analyse de sentiments, une partie des corpus ont un thème similaire, comme la haine, la misogynie, ou l'agressivité. Enfin, les autres jeux de données ont des thématiques plus classiques, comme la détection du sentiments, ou de l'ironie. Dans cette partie, nous présentons chacun de ces jeux de données, dont la répartition est récapitulée dans le tableau 3.1.

HatEval [10] correspond à la tâche 5 du concours SemEval 2019. Cette tâche consiste à détecter la haine contre les immigrants et les femmes dans les données Twitter. Les tweets ont été principalement collectés entre juillet et septembre 2018 en utilisant trois approches différentes : l'utilisation de mots-clés pour filtrer les flux Twitter, l'utilisation de l'historique des utilisateurs postant des messages haineux précédemment détectés, et la surveillance des comptes qui pourraient être victimes de haine. Il existe deux ensembles de données : en Anglais et en Espagnol. Ayant participé à cette tâche SemEval (notre participation est décrite dans le Chapitre 5), nous avons exploré les corpus des deux langues. Cependant, pour nos expérimentations, nous avons uniquement utilisé le jeu de données en Anglais composé de 13 000 tweets (5 790 tweets non haineux et 4 210 tweets haineux pour la partie train).

OffensEval [170] est la tâche 6 de la compétition SemEval 2019. L'objectif est de classer un jeu de données de 14 100 tweets (13 240 pour l'apprentissage et 860 pour la validation) entre offensif et non offensif. Les tweets ont été collectés en utilisant des mots-clés [169] et annotés grâce au *CrowdSourcing* [6]. Le jeu de données d'entraînement contient un total de 13 240 tweets : 8 840 non-offensifs (66,8 %) et 4 400 offensifs (33,2 %).

IberEval [36] est le jeu de données utilisé dans le cadre de la compétition AMI (*Automatic Misogyny Identification*) en 2018. Pour cette compétition, l'objectif était de détecter la misogynie dans les tweets. Le jeu de données est composé de 1 683 tweets

5. évalue la capacité d'un modèle de classification à prédire efficacement les individus positifs, en faisant un compromis entre la *precision* (taux de prédictions positives correctes) et le *recall* (taux de positifs correctement prédits)

6. Consiste à faire appel au grand public pour participer à la création d'un élément. 360-Webmarketing

non misogynes, et de 1 568 tweets qui contiennent de la misogynie. Ce jeu de données est le seul de notre ensemble de corpus qui ne provient pas de SemEval.

Pour le jeu de donnée de la tâche 6a de SemEval 2016 [100], nous avons utilisé le jeu d'entraînement, qui contient 2 914 tweets, répartis en trois classes : 1 762 tweets négatifs, 963 tweets positifs et 189 qui correspondent à la classe "autre". Le but de la tâche ici est de faire de la détection de posture sur les cinq cibles suivantes : 'Athéisme", "Le changement climatique est une préoccupation réelle" [7], "Mouvement féministe", "Hillary Clinton" et "Légalisation de l'avortement".

Le corpus de la tâche 4 de SemEval 2017 [127] est une reprise de la tâche 4 de SemEval 2016 [103]. Elle comporte cinq sous-tâches, mais nous n'avons utilisé que la sous-tâche A, qui vise à effectuer une analyse de sentiments dans des tweets sur une échelle de trois niveaux (positif, neutre et négatif). La tâche est disponible en deux langues : anglais et arabe. Nous avons travaillé avec le jeu de données anglais, qui contient un total de 62 617 tweets (50 333 pour le jeu de données d'entraînement et 12 284 pour le jeu de données de test). Ces données correspondent à différents thèmes d'entités nommées (par exemple Donald Trump, iPhone...), entité géopolitiques (Alep, Palestine...), ainsi que d'autres thématique comme les réfugiés syriens, le *Dakota Access Pipeline*, les médias occidentaux, le contrôle des armes à feu ou encore le végétarisme. Elles ont été annotées grâce au *CrowdSourcing*.

Enfin, le dernier jeu de données de notre ensemble est celui utilisé pour la tâche 3 de SemEval 2018 [150]. Le corpus contient 3 000 tweets, avec 2 396 tweets ironiques et 604 autres non ironiques. Cette tâche a pour objectif la détection d'ironie, qui est une branche de l'analyse de sentiments.

Nous pouvons constater que les bases de données existantes dans la littérature présentent des situations diverses. Par exemple, la tâche 9 de SemEval 2015 comporte peu de tweets, ce qui rend difficile l'apprentissage avec des modèles Transformer. Les jeux de données des tâches 10 et 11 de SemEval 2015 sont également déséquilibrées. Tous ces éléments compliquent l'utilisation de modèles d'apprentissage profond. Ainsi, nous avons sélectionné les bases décrites ci-dessus, car elles présentent l'avantage d'être diverses et variées, tout en permettant l'utilisation des modèles à l'état de l'art.

Jeux de données	Classe 1	Classe 2	Classe 3
HateEval	4 210 haineux	5 790 non haineux	-
OffensEval	4 400 agressifs	8 840 non agressifs	-
IberEval	1 568 misogynes	1 683 non misogynes	-
SE2016 tâche 6	963 positifs	1 762 négatifs	189 "autres"
SE2017 tâche 4a	19 902 positifs	7 840 négatifs	22 591 neutres
SE2018 tâche 3a	2 396 ironiques	604 non ironiques	-

TABLE 3.1 – Distribution des classes des jeux de données

7. « *Climate Change is a Real Concern* »

3.3 Quelles sont les bonnes pratiques pour faire de l'analyse de sentiments avec des ressources limitées ?

Dans cette partie, nous commençons par comparer les deux tailles de modèles, BERT-base et BERT-large. Nous poursuivons en testant les deux modèles BERT sur l'oubli catastrophique, en proposant une solution lorsque le modèle y est sensible, puis nous faisons un choix sur le type d'apprentissage à favoriser. Enfin, nous terminons par voir les spécificités du vocabulaire Twitter, ainsi que les pré-traitements à leur appliquer pour faire de la classification de sentiments.

3.3.1 Choix de la taille du modèle

Le modèle BERT existe en différentes tailles : BERT-base et BERT-large. Les différences entre ces deux modèles sont le nombre de couches (12 pour BERT-base, 24 pour BERT-large), le nombre de têtes d'attention (12 têtes d'attention pour BERT-base, 16 têtes d'attention pour BERT-large) et le nombre de paramètres (110 millions de paramètres pour BERT-base, 340 millions de paramètres pour BERT-large). Avec plus de paramètres et plus de têtes d'attention, nous pouvons nous attendre à ce que BERT-large ait de meilleures performances de prédiction que le plus petit modèle, BERT-base. Mais les modèles plus grands sont également plus longs et plus difficiles à entraîner et à affiner. Notre objectif ici est donc de comparer les scores, les écarts-types et les temps de *fine-tuning* des deux modèles afin de déterminer si l'utilisation de BERT-large sur les tweets améliore réellement la classification, même si les ressources sont limitées.

Dans le tableau 3.2, les résultats du score f1 moyen, ainsi que les écarts-types, sont présentés pour le *fine-tuning* de BERT-base et BERT-large. Puisque nous sommes plus particulièrement intéressés par les ressources et le temps de calcul, le tableau 3.2 présente également le temps d'affinage et d'entraînement pour chaque taille de modèle, sur chacun des jeux de données.

	BERT-base			BERT-large		
	F1	std	temps (sec)	F1	std	temps (sec)
HateEval	83,08	1,01	43 248	82,95	1,08	145 180
OffensEval	78,33	0,86	57 004	78,5	0,93	192 189
IberEval	80,32	2,47	14 106	79,91	2,25	46 857
SE2016 tâche 6	68,17	3,43	12 567	66,05	3,4	42 379
SE2017 tâche 4a	72,77	1,07	89 176	73,35	1,04	299 630
SE2018 tâche 3a	71,9	1,83	16 445	71,68	2,53	55 426

TABLE 3.2 – Scores F1 et écarts-types (std) pour BERT-base et BERT-large

Contrairement à ce que l'on pourrait penser, les résultats du tableau 3.2 montrent que l'utilisation de BERT-large n'améliore pas les performances du modèle. Les scores f1 de

BERT-large sont, en effet, similaires à ceux de BERT-base. L'utilisation du plus grand modèle permet au mieux de gagner 0,58 points de score F1 pour le jeu de données de SemEval 2017 tâche 4a, mais fais également perdre jusqu'à 2,12 points de score F1 pour le corpus de SemEval 2016 tâche 6. Ces résultats sont en contradiction avec l'intuition que l'on peut avoir sur les grands modèles profonds : plus le modèle a de paramètres, plus il est performant. En effet, dans ce cas, non seulement l'utilisation du plus grand modèle n'améliore pas les scores, mais elle augmente fortement le temps de *fine-tuning* (en moyenne 3,36 fois plus long). La figure 3.2 nous montre l'évolution du score F1 moyen au fil des epochs. Chaque courbe correspond aux différents taux d'apprentissage et états des couches (gelées ou non). Comme nous pouvons le voir sur la figure 3.2, le score F1 continue d'augmenter pour le jeu de données SemEval 2016 tâche 6. Ce jeu de données est le plus petit dans notre ensemble de corpus, donc cela pourrait indiquer que la quantité de données n'est pas suffisante pour affiner correctement le grand modèle. Au contraire, le score F1 pour le jeu de données SemEval 2017 tâche 4a, qui est le plus grand de notre ensemble, s'est stabilisé avant la dernière epoch, ce qui peut être identifié comme une quantité suffisante de données.

Comme nous venons de le voir, les résultats présentés dans le tableau 3.2 sont contraires à l'intuition sur les grands modèles avec beaucoup de paramètres. Cela nous amène à nous demander si les performances de BERT-large ne sont pas freinées par l'oubli catastrophique.

3.3.2 Gérer l'oubli catastrophique

Dans la partie précédente, nous avons pu voir que l'apprentissage de BERT-large semblait freiné par l'oubli catastrophique [97] lors de son *fine-tuning*. Afin de vérifier cela, nous proposons de mener une série d'expérimentation, à la fois sur BERT-base et BERT-large afin de vérifier la perte de connaissances. Pour cela, nous avons repris les solutions à l'oubli catastrophique décrites dans la partie 3.1.1, c'est-à-dire le gel de certaines des couches du modèle [80], ainsi que l'utilisation d'un taux d'apprentissage faible [105]. Nous avons donc comparé trois différents taux d'apprentissage, un taux élevé ($5e^{-5}$), un taux moyen ($3e^{-5}$) et un taux faible ($1e{-}5$), que nous avons combinés ou non au gel de certaines couches du modèle (toutes, sauf le dernier quart).

Notre modèle a donc été affiné dans ces différentes configurations, et nous avons évalué le modèle pendant le *fine-tuning*. Nous avons mesuré le score F1 et la fonction de perte d'apprentissage plusieurs fois au cours de chaque epoch, et nous avons reporté les résultats sur les figures 3.2 et 3.3 pour mieux observer l'évolution du score f1 du modèle et de sa fonction de perte.

La figure 3.2 présenté dans la partie précédente, ainsi que la figure 3.3, représentent les conditions de *fine-tuning* pour chaque jeu de données avec le modèle BERT-large. Chaque courbe correspond à l'évolution de la fonction de perte d'entraînement avec ou sans couches gelées et avec un taux d'apprentissage faible, moyen ou élevé. Nous pouvons voir que sans gel des couches et avec le taux d'apprentissage moyen et élevé, les scores f1 n'augmentent pas pendant l'apprentissage (figure 3.2), et les fonctions de perte ne diminuent pas non plus : le modèle a des difficultés à apprendre (fi-

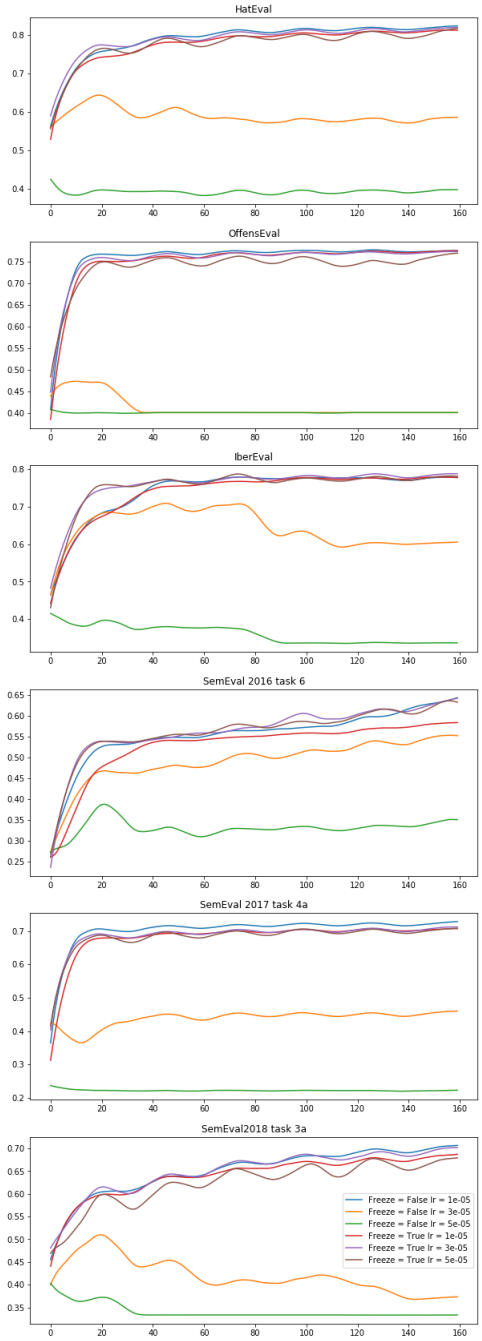

FIGURE 3.2 – Scores F1 moyen pour chaque jeu de données, avec chacune des configuration, pour le modèle BERT-large. Pour chaque jeu de données, huit mesures ont été prise par epoch (sept pendant et une après le *fine-tuning*), pendant 20 epochs

gure 3.3).

Les scores du modèle pour chaque taux d'apprentissage et état des couches sont également indiqués dans le tableau 3.3 pour BERT-base et tableau 3.4 pour BERT-large, avec chaque jeu de données.

Pour le modèle BERT-base (dans le tableau 3.3), le plus petit des deux modèles, il ne semble pas y avoir de sensibilité à l'oubli catastrophique. En effet, les scores F1 et les écarts-types sont stables, quel que soit le taux d'apprentissage, ou le statut des couches (gelées ou non).

Pour BERT-large en revanche (tableau 3.4), nous pouvons constater que sans gel des couches, plus le taux d'apprentissage est haut, plus le modèle a du mal à apprendre. En effet, avec les taux d'apprentissage moyen et élevé, les scores F1s sont extrêmement bas et les écarts-types très haut. Dans certains cas, le modèle ne prédis qu'une seule classe quelque soit l'epoch et le découpage, il n'apprend donc pas du tout. Pour OffensEval par exemple, nous pouvons voir un score F1 moyen de 40,04, mais surtout un écart-type de 0. Cela est également visible dans les courbes en figure 3.2 et figure 3.3 : les scores F1 (figure 3.2) n'augmentent pas au fil des epochs, et les fonctions de perte (figure 3.3) ne diminuent pas ou oscillent. En revanche, ce problème disparaît en gelant les couches, ou en utilisant le taux d'apprentissage le plus faible. En effet, nous pouvons constater que les courbes des fonctions de perte ont une nette tendance à la baisse (figure 3.3) et que les scores F1 augmentent avant de se stabiliser (figure 3.2). L'utilisation du taux d'apprentissage faible permet d'obtenir un gain allant jusqu'à 49,35 points de score F1 pour le jeu de données SemEval 2017 tâche 4a. De manière général, le taux d'apprentissage faible permet une augmentation de 35,87 points de score F1 en moyenne comparé au taux d'apprentissage élevé. Également, appliquer le gel des couches permet d'obtenir des performances similaires à celles du taux d'apprentissage faible : les courbes ont le même comportement, la fonction de perte diminue (figure 3.3), le score augmente (figure 3.2) et les scores F1 sont cohérents (tableau 3.4).

Si utiliser un faible taux d'apprentissage a un impact plus important sur l'amélioration des performances, geler les couches du modèle BERT-large permet d'obtenir d'autres avantages : en plus de réduire la plupart des écarts-types du modèle, cela entraîne une réduction du temps d'affinage du modèle, ce qui est avantageux lorsque nous n'avons à disposition qu'un unique GPU. Pour nos plus grands jeux de données (OffensEval et SemEval 2017 tâche 4a), dont le *fine-tuning* prend plus de deux jours, cela permet donc de réduire considérablement le temps d'attente.

Cependant, bien que les solutions décrites ci-dessus permettent de réduire, voir de faire disparaître la sensibilité de BERT-large à l'oubli catastrophique, cette approche n'est pas optimale. En effet, l'utilisation du grand modèle reste trop contraignante comparée à l'utilisation de BERT-base. De plus, comme nous avons pu le constater précédemment, les performances de BERT-large ne sont pas supérieur à celle du modèle plus petit (BERT-base), alors qu'il est plus long à affiner que ce dernier. Ainsi, au vu de la sensibilité du modèle BERT-large à l'oubli catastrophique, de ses performances au final très proches de BERT-base et de son temps de *fine-tuning*, nous recommandons

donc l'utilisation du modèle plus petit, BERT-base. Nous poursuivons ces expérimentations en nous concentrant principalement sur ce modèle. Nous retiendrons tout de même les solutions permettant de limiter l'oubli catastrophique, qui nous servirons plus tard pour nos expérimentations avec le modèle BERTweet [104].

	Sans gel des couches						Avec gel des couches					
	$1e^{-5}$		$3e^{-5}$		$5e^{-5}$		$1e^{-5}$		$3e^{-5}$		$5e^{-5}$	
	F1	std	F1	std	F1	std	F1	std	F1	std	F1	std
HateEval	82,63	1,04	**83,08**	1,01	82,77	1,66	81,76	0,94	82,59	1,14	**82,77**	1,02
OffensEval	78,19	1,05	**78,33**	0,86	77,65	0,84	77,77	0,81	77,51	0,63	77,69	0,72
IberEval	79,55	2,46	**80,32**	2,47	80,18	1,55	78,59	2,85	79,36	2,62	**79,64**	2,85
SE2016 tâche 6	66,63	2,97	**68,15**	3,43	67,65	2,56	60,00	2,9	**67,55**	2,29	66,9	2,41
SE2017 tâche 4a	72,47	1,19	**72,77**	1,07	72,32	1,08	**71,75**	1,34	71,73	1,35	71,64	1,24
SE2018 tâche 3a	71,99	1,98	71,9	1,83	**72,46**	1,98	69,93	2,47	70,86	2,5	**71,74**	2,24
Rang moyen	3,17		1,33		2,67		5,17		4,83		3,83	

TABLE 3.3 – Performance de la BERT-base sur chaque jeu de données, pour les trois taux d'apprentissage, et avec ou sans couches gelées.

	Sans gel des couches						Avec gel des couches					
	$1e^{-5}$		$3e^{-5}$		$5e^{-5}$		$1e^{-5}$		$3e^{-5}$		$5e^{-5}$	
	F1	std	F1	std	F1	std	F1	std	F1	std	F1	std
HateEval	**82,95**	1,08	67,8	19,15	40,11	10,26	81,89	1,00	**82,65**	1,04	81,78	0,96
OffensEval	**78,5**	0,93	50,75	16,38	40,04	0,0	**78,14**	0,75	78,13	0,87	77,56	1,07
IberEval	**79,91**	2,25	76,67	6,20	46,37	14,07	79,94	2,27	79,07	2,27	**80,0**	2,19
SE2016 tâche 6	**66,05**	3,4	58,04	16,84	44,94	13,59	60,09	2,73	**65,66**	3,96	62,27	2,9
SE2017 tâche 4a	**73,35**	1,04	50,98	23,48	24,0	5,15	**71,79**	1,07	71,39	0,98	71,64	0,89
SE2018 tâche 3a	**71,68**	2,53	56,77	12,8	41,78	11,08	69,75	1,91	**71,58**	1,9	69,53	3,03
Rang moyen	1,33		5		6		2,67		2,83		3,17	

TABLE 3.4 – Performance de la BERT-large sur chaque jeu de données, pour les trois taux d'apprentissage, et avec ou sans couches gelées.

3.3.3 Choisir le type d'apprentissage pour BERT-base

Après avoir trouvé la taille idéale pour notre modèle, BERT-base, nous avons voulu déterminer si l'utilisation du *fine-tuning* était suffisante, ou si le modèle avait besoin d'être encore plus spécialisé sur les données Twitter. L'apprentissage d'un modèle de langage à partir de zéro sur des tweets n'est pas réalisable avec notre protocole expérimental (c'est-à-dire avec un seul GPU). Ainsi, nous sommes légèrement sortis du protocole que nous avons fixé afin de pouvoir poursuivre le pré-apprentissage de BERT-base. Ce nouveau cadre constitue un compromis entre l'utilisation du *fine-tuning* et l'apprentissage d'un modèle de zéro. Nous avons donc utilisé un modèle BERT-base préalablement entraîné et nous avons poursuivi son apprentissage en utilisant des données Twitter. Poursuivre le pré-entraînement d'un modèle permet de mieux adapter ses poids qu'avec un simple affinage. Pour ce faire, nous avons dû utiliser un cluster de 4 GPU NVIDIA V100, ce qui reste tout de même éloigné de la puissance de calcul utilisée par Google pour BERT. Le modèle a alors subit un entraînement supplémentaire sur un jeu de données de 93 millions tweets pendant deux semaines. Les résultats dans le tableau 3.5 nous permettent de mesurer l'impact de la poursuite du pré-apprentissage sur les tweets et de comparer cette approche avec le *fine-tuning*.

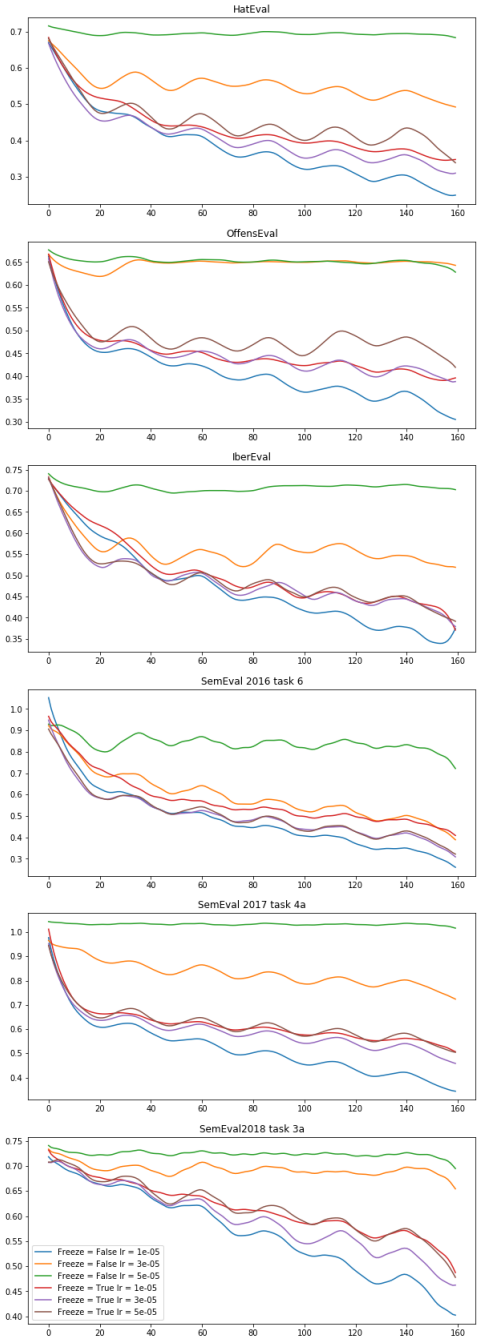

FIGURE 3.3 – Fonction de perte d'apprentissage moyenne pour chaque jeu de données, avec chacune des configuration, pour le modèle BERT-large. Pour chaque jeu de données, huit mesures ont été prise par epoch (sept pendant et une après le *fine-tuning*), pendant 20 epochs

Les résultats obtenus (tableau 3.5) montrent que les scores F1 de la poursuite du pré-entraînement surpassent ceux de l'affinage pour deux de nos jeux de données. Cette méthode permet de gagner jusqu'à 1,91 points de score f1 pour le jeu de données SemEval 2018 tâche 3a et 0,71 points pour SemEval 2017 tâche 4a. Les écarts-types ont également été réduits pour certains des jeux de données : pour le corpus SemEval 2016 tâche 6, ils ont été divisés par 4,45. Cependant, pour la majorité des corpus utilisés, les résultats sont inférieurs pour la poursuite du pré-apprentissage.

Si intuitivement, poursuivre l'apprentissage d'un modèle parait être la meilleure solution car cela permet de mieux adapter le modèle aux données en comparaison au *fine-tuning*, BERT-base obtient des résultats supérieurs. En effet, au vu des scores, il semblerait que le pré-entraînement ne soit pas suffisamment poursuivi pour avoir un véritable impact sur le modèle. Ainsi, cette solution correspondant à la demie-mesure entre le *fine-tuning* et l'apprentissage de zéro, n'est pas suffisante. De plus, poursuivre l'apprentissage du modèle a nécessité de sortir du cadre fixé au départ, c'est-à-dire l'utilisation d'un unique GPU. Or, moins il y a de GPU disponibles, plus l'apprentissage prend du temps. L'application de cette méthode avec un seul GPU prendra donc plus de deux semaines, et au contraire, pour réduire encore le temps de pré-entraînement, plus de GPU seront nécessaires.

Étant donné que poursuivre le pré-apprentissage est insuffisant, en plus de nécessiter davantage de GPU et de ressources en temps, la solution la plus adaptée à notre situation est donc le *fine-tuning* du modèle BERT-base. Cependant, comme nous l'avons vu précédemment, les tweets sont composés d'un vocabulaire particulier. Le manque de ressources ne nous permet pas de réaliser un apprentissage de zéro, et nous avons choisis de ne pas poursuivre l'apprentissage de BERT-base dû aux performances de cette approche. Puisque les méthodes permettant d'adapter un modèle générique ne sont pas suffisantes, il est nécessaire de s'intéresser à la normalisation du vocabulaire des tweets afin de ne pas impacter les performances du modèles. Nous verrons dans la partie suivante comment gérer ces particularités des tweets pour l'analyse du sentiment sur Twitter en nous concentrant d'abord sur la normalisation des tweets pour le *fine-tuning* de BERT, puis nous comparons cette approche à l'utilisation d'un modèle déjà adapté au vocabulaire de Twitter, BERTweet [104].

	BERT-base			Poursuite du pré-apprentissage de BERT		
	F1	std	time (sec)	F1	std	time (sec)
HateEval	83,08	1,01	43 248	83,26	0,75	≈ 2 weeks
OffensEval	78,33	0,86	57 004	78,11	0,89	
IberEval	80,32	2,47	14 106	79,82	1,76	
SE2016 tâche 6	68,17	3,43	12 567	66,01	0,77	
SE2017 tâche 4a	72,77	1,07	89 176	73,48	1,07	
SE2018 tâche 3a	71,9	1,83	16 445	73,81	1,98	

TABLE 3.5 – Scores F1 et écarts-types (std) pour chacun des modèles utilisés

3.3.4 Traiter le vocabulaire spécifique de Twitter

Le vocabulaire Twitter est très particulier. En effet, les tweets peuvent contenir des hashtags (ou « mot-dièse »), des mentions d'utilisateurs, des URL, des marqueurs de retweets, ou même des emojis. Ils peuvent également contenir des fautes d'orthographe ou de grammaire, ou du sarcasme. Nous pensons que tous ces éléments peuvent impacter l'analyse de sentiments dans les tweets s'ils sont omis ou ignorés. C'est pourquoi nous proposons dans cette partie de voir comment gérer les spécificités de Twitter. Nous parlerons tout d'abord du traitement des mentions, hashtags et URL, puis nous voyons comment l'utilisation d'un modèle spécialisé permet de faciliter le traitement des particularités de Twitter. Enfin, nous terminons par une discussion du cas particulier des emojis.

3.3.4.1 Normalisation des tweets : que faire des mentions, hashtags et URL

Comme nous l'avons vu dans le Chapitre 2, les tweets contiennent des caractéristiques spéciales qui doivent être normalisées afin que les tweets soient traités correctement. Le modèle BERT se charge de prendre en compte certaines de ces caractéristiques, comme les fautes d'orthographe, gérées par l'utilisation de WordPiece [8] [163] afin de découper le texte. De plus, avec WordPiece, l'absence d'espace entre le texte et les caractères spéciaux ne gêne pas le tokenizer de BERT. Cependant, certaines spécificités du vocabulaire Twitter ne sont pas prises en compte par le modèle, et nécessitent des pré-traitements.

Le consensus de la littérature concernant les URL et les mentions d'utilisateur est le remplacement par un tag [46, 154]. Il est également nécessaire d'adapter le tokenizer afin qu'il puisse prendre en compte les nouveaux *tokens*, tels que les tags. Ce prétraitement permet à la fois de normaliser et d'anonymiser les jeux de données. Mais cela augmente également le temps d'apprentissage, car le modèle doit apprendre la représentation de nouveaux *tokens*. Concernant les hashtags, ils peuvent être laissés et donc non remplacés, car là encore WordPiece [163] peut naturellement les prendre en compte tels quels. Il n'est donc pas nécessaire de les découper en mot au préalable. Ceci permet au modèle de traiter le texte contenu dans les hashtags, sans aucune perte d'information [104]. Les *tokens* spéciaux restant non traités seront considérés comme ne faisant pas partie du dictionnaire et seront représentés avec le *token* « UNK ». En effet ceux ci ne sont pas utiles à la classification des tweets et peuvent même la perturber.

Dans la partie suivante, nous démontrons comment l'utilisation d'un modèle spécialisé, lorsqu'il est disponible, permet de faciliter la normalisation des tweets, tout en améliorant la classification.

8. Permet un découpage non pas au niveau des mots, mais au niveau du sous-mot, cette méthode de tokenisation est inspirée du *Byte Pair Encoding* (BPE)[41]

3.3.4.2 Utilisation d'un modèle Twitter : BERTweet

Dans les parties précédentes, nous avons décrit deux solutions permettant de faire de l'analyse de sentiments avec des ressources limitées. Tout d'abord, il est possible d'affiner un modèle générique, comme BERT. Ensuite, le pré-apprentissage d'un modèle générique peut également être poursuivi. Malgré une meilleure adaptation du modèle aux tweets, cette méthode est insuffisante. De plus, poursuivre le pré-apprentissage est plus long que le *fine-tuning* d'un modèle, et nécessite plus de puissance de calcul. Ainsi, nous avons pu voir que les méthodes classiques pour adapter un modèle de langage générique au traitement de tweets n'étaient pas suffisantes. Nous souhaitons donc confronter ces méthodes à l'utilisation d'un modèle spécialisé, entraîné sur des données issues de Twitter. Dans cette partie, nous comparons le *fine-tuning* de BERTweet, fondé sur le modèle BERT-base, mais entraîné sur des tweets, au *fine-tuning* du modèle BERT-base classique.

Pour BERTweet [104], (Nguyen et al., 2020) normalisent les tweets en remplaçant les mentions d'utilisateurs et les URL par un tag. De plus, les données subissent également une étape de démojisation, afin de remplacer chaque emojis présent dans les tweets par son texte.

BERTweet étant fortement inspiré de BERT, avec la même architecture que BERT-base, nous avons voulu tester la sensibilité du modèle à l'oubli catastrophique. Nous avons donc reproduit les expérimentations précédemment décrites afin de vérifier si BERTweet était sujet à l'oubli catastrophique, et si les solutions identifiées pour BERT-large étaient compatibles. En faisant cela, nous avons pu découvrir que le modèle Twitter était, comme BERT-large, sujet à la perte de connaissances. Cette fois encore, l'oubli catastrophique a pu être évité grâce aux solutions appliquées à BERT-large, c'est-à-dire grâce à l'utilisation d'un faible taux d'apprentissage et au gel des couches. Les trois approches peuvent donc maintenant être comparées de manière équivalente.

Les résultats concernant BERTweet sont reportés dans le tableau 3.6, de même que ceux de l'affinage de BERT-base, et de la poursuite du pre-apprentissage de BERT-base, ce afin de faciliter la comparaison. Nous pouvons voir que les performances de BERTweet surpassent celles des autres approches dans la grande majorité des cas, en particulier pour les jeux de données SemEval 2017 tâche 4a et SemEval 2018 tâche 4a. En effet, pour SemEval 2018 tâche 3a, l'utilisation de BERTweet permet de gagner plus de 5 points de score F1 comparé au *fine-tuning* de BERT-base, et plus de 3 points comparé à la poursuite du pré-apprentissage du même modèle. Ceci nous permet de voir que l'utilisation d'un modèle spécialisé est bien plus adaptée au traitement des tweets. En effet, si le *fine-tuning* permet d'adapter un modèle de langage générique en mettant à jour ses poids, un modèle de langage Twitter possède des poids déjà adaptés aux données et nécessite seulement d'être adapté à la tâche. De plus, les caractères spéciaux et le vocabulaire spécifique lié à Twitter seront absents d'un modèle de langage générique si celui-ci n'a jamais été entraîné sur des tweets.

BERTweet étant un modèle spécialisé pour le traitement des tweets, il est donc adapté au vocabulaire particulier de Twitter. Concernant les emojis, BERTweet applique une

étape de démojisation à chacun des tweets. Ainsi, nous cherchons à déterminer si l'application de ce traitement en amont du *fine-tuning* de BERT-base permet de combler les écarts de performances avec BERTweet. Dans la partie suivante, nous comparons donc le modèle BERT-base affiné avec et sans étape de démojisation, ainsi que BERTweet.

	BERT-base		Poursuite du pré-apprentissage de BERT		BERTweet	
	F1	std	F1	std	F1	std
HateEval	83,08	1,01	83,26	0,75	**83,84**	0,86
OffensEval	78,33	0,86	78,11	0,89	**78,74**	1,01
IberEval	**80,32**	2,47	79,82	1,76	79,62	2,18
SE2016 tâche 6	68,17	3,43	66,01	0,77	**69,99**	2,51
SE2017 tâche 4a	72,77	1,07	73,48	1,07	**75,24**	1,04
SE2018 tâche 3a	71,9	1,83	73,81	1,98	**77,01**	1,45

TABLE 3.6 – Scores F1 et écarts-types (std) moyen pour BERTweet [104], en comparaison au *fine-tuning* de BERT-base ainsi qu'à la poursuite du pré-apprentissage de BERT-base

3.3.4.3 Traitement des emojis

Parmi les particularités du vocabulaire Twitter, nous pouvons également retrouver les emojis. Si certaines études les suppriment tout simplement après avoir utilisé la supervision distante [46, 75, 154], il est utile de conserver une trace des emojis et de l'information qu'ils contiennent. Pour les traiter, nous avons utilisé la démojisation, également appliquée à BERTweet [104]. La démojisation consiste à remplacer un emoji par son texte. Pour cela, nous nous sommes servi de l'API *Emoji*[9]. Notre objectif ici est donc de comparer le *fine-tuning* de BERT-base avec et sans démojisation, ainsi que BERTweet, et ce afin de déterminer si ce traitement explique les meilleures performances du modèle spécialisé.

Dans le tableau 3.7, les scores F1 moyens ainsi que les écarts-types sont présentés pour le *fine-tuning* de BERT-base ainsi que pour la poursuite de son pré-apprentissage avec démojisation, sur les quatre jeux de données qui contiennent des emojis. Notre objectif étant de tester l'efficacité de la démojisation, les résultats de chaque modèle sont comparés à ceux sans démojisation (contenus dans le tableau 3.6).

Mis à part SemEval 2018 tâche 3a, nous pouvons constater que la démojisation n'améliore pas les performance de BERT-base, que ce soit pour le *fine-tuning*, ou après avoir poursuivi son pré-apprentissage. Pour trois de nos quatre jeux de données qui contiennent des emojis, les gains sont toujours négatifs. Concernant le jeu de données SemEval2018 tâche 3a, la poursuite du pré-apprentissage de BERT-base est amélioré par la démojisation. En effet, le score F1 moyen est augmenté de 0,85 points. Ce gain est dû à la spécificité de la tâche, la détection d'ironie, pour laquelle le traitement des emojis joue un rôle important. En revanche, il faut également noter que l'écart-type pour le modèle dont le pré-apprentissage a été poursuivi a augmenté de 1 point pour

9. https://pypi.org/project/emoji/

ce corpus, passant de 1,98 à 2,92. Nous pouvons donc voir que le modèle a perdu en généralisation.

Par conséquent, nous pouvons constater que pour l'utilisation de modèles génériques tels que BERT, qui n'ont pas appliqué de démojisation lors de leur pré-apprentissage, remplacer les emojis par leur texte n'est pas efficace. En effet, si le modèle n'a pas appris à traiter les emojis remplacés durant son étape de pré-entraînement, il ne peut pas correctement les gérer.

Si la démojisation n'améliore pas la classification, d'autres solutions existent pour le traitement des emojis, tels que le remplacement par leur description [139], ou encore l'utilisation d'un *embedding* d'emojis [8, 35]. Ce genre de solution a été le sujet d'autres expérimentations, dont nous parlerons Chapitre 4. Dans la partie suivante, nous discutons des résultats obtenus.

	BERT-base				Poursuite du pré-apprentissage de BERT			
	F1	Gain	std	Gain	F1	Gain	std	Gain
HateEval	82,93	-0,15	0,86	-0,15	82,88	-0,38	0,78	+0,03
OffensEval	77,99	-0,34	0,96	+0,1	78,13	+0,02	1,03	+0,14
IberEval	79,4	-0,92	1,58	-0,89	79,4	-0,42	2,04	+0,28
SE2018 tâche 3a	72,75	+0,85	1,59	-0,24	74,9	+1,09	2,92	+0,94

TABLE 3.7 – Scores F1 et écarts-types (std) avec démojisation pour les jeux de données avec emojis, comparés aux résultats sans démojisation

3.3.5 Résultats

La limite en puissance de calcul, mémoire et données rend plus difficile la tâche d'analyse de sentiments. En effet, tout le monde ne peut pas avoir accès à plusieurs GPU, il est donc fréquent de ne pas avoir les ressources nécessaires au pré-apprentissage d'un modèle Transformer. Pour palier le manque de GPU, une série d'expériences a été menée dans le but de définir des bonnes pratiques, et pouvoir tout de même faire de l'analyse de sentiments Twitter dans ces conditions. Ces recommandations sont valables pour les situations où les textes à traiter ne sont pas écrits dans un langage courant et peuvent être résumées comme ceci :

— favoriser l'utilisation d'un modèle spécialisé lorsqu'il celui-ci est disponible. Un modèle spécialisé est un modèle ayant été pré-entraîné sur le même type de données (Twitter, Reddit, news, etc.) ou la même thématique (domaine médical, technologie, misogynie, etc) que les données cibles.

— si aucun modèle spécialisé n'est disponible , un modèle générique peut être utilisé après avoir été affiné, en respectant certaines conditions dont nous discuterons ci-dessous.

BERTweet semble donc être la solution parfaite pour l'analyse de sentiments Twitter avec des ressources limitées, puisque le *fine-tuning* ne nécessite que peu de temps et de GPU. Cette approche a de meilleurs résultats que l'affinage des modèles génériques BERT-base ou BERT-large, et surpasse même la poursuite du pré-apprentissage tout

en ayant un temps d'adaptation plus court. De plus, ce modèle est facile d'accès, grâce à HuggingFace [10].

Cependant, si un modèle Twitter est disponible pour l'architecture de BERT, ce n'est pas le cas pour tous les modèles Transformer. De plus, au vu de la vitesse d'apparition de nouvelles architectures fondées sur le Transformer, il faudra attendre avant que des versions spécialisées ne soient publiées. Comme nous l'avons dit précédemment, le pré-entraînement d'un modèle est coûteux, cela nécessite beaucoup de temps et de GPU, qui sont difficilement accessibles. Dans ce cas, lorsqu'aucun modèle Twitter n'est disponible, un modèle générique peut être affiné sur des tweets.

Si l'intuition veut que BERT-large obtienne de meilleurs scores que BERT-base, ce ne fut pas le cas lors de nos expérimentations. En effet, le plus grand des deux modèles a obtenu des performances similaires, voire inférieures à BERT-base, tout en ayant un temps de *fine-tuning* bien plus long que le modèle base. Ainsi, nous pouvons considérer que le modèle BERT-base affiné est plus efficace pour faire l'analyse de sentiments en ayant accès qu'à peu de ressources. C'est donc ce modèle, et non BERT-large qui doit être favorisé pour cette tâche. Si toutefois, pour une quelconque raison, de l'analyse de sentiments doit être effectuée avec BERT-large, le *fine-tuning* doit être utilisé avec précaution, car il peut engendrer de l'oubli catastrophique. Un taux d'apprentissage faible et un gel des premières couches du modèle permettent de limiter ce problème.

L'affinage de BERT n'est pas la seule solution disponible lorsque il est impossible d'apprendre un modèle de zéro, il est également possible de poursuivre le pré-apprentissage du modèle. Malheureusement, poursuivre le pré-apprentissage de BERT-base reste trop long et demande beaucoup de ressources, il ne peut donc pas être effectué en ayant à disposition qu'un unique GPU. Cette solution n'est donc pas adaptée à notre problématique de classification de sentiments avec ressources limitées.

Puisque les modèles génériques ne sont pas adaptés au vocabulaire de Twitter, les tweets doivent être normalisés. La littérature recommande de remplacer les mentions d'utilisateurs et les URL par des tags, mais de ne pas appliquer de traitement aux hashtags, qui seront traités comme le reste du texte grâce à WordPiece. Suite à ces pré-traitements, le tokenizer doit également être adapté en ajoutant les nouveaux tags à la liste des tokens. Cette adaptation augmente le temps de traitement, puisqu'une représentation doit être construite pour les nouveaux tokens. Enfin, si la démojisation est automatiquement effectuée par le modèle BERTweet, ce traitement des emojis n'est pas efficace lorsqu'un modèle générique est affiné ou que son apprentissage est poursuivi. Dans l'attente d'une meilleure solution, décrite dans le Chapitre 4, nous recommandons pour le moment de supprimer les emojis afin qu'ils évitent de perturber le modèle de langage, ou encore de les laisser tel quel dans le texte, où ils ne seront pas pris en compte car hors dictionnaire.

10. https://huggingface.co/transformers/model_doc/bertweet.html

3.4 Conclusion

Cette série d'expérimentations nous a permis d'identifier les bonnes pratiques pour faire de l'analyse de sentiments avec des ressources limitées. Nous nous sommes intéressés à la taille du modèle générique BERT, à son type d'apprentissage ainsi qu'aux traitements des spécificités de Twitter dans le cadre du *fine-tuning* de ce modèle. Nous avons également vu qu'il pouvait exister des modèles spécialisés, c'est-à-dire entièrement entraînés sur des données similaires aux données cible, comme BERTweet, qui a été pré-appris sur des tweets.

L'utilisation de BERTweet est la meilleure solution pour faire de l'analyse de sentiments sur des données issues de Twitter avec le modèle Transformer BERT. En effet, le modèle étant spécialisé, il nécessite d'être adapté seulement à la tâche, car ses poids et ses *tokens* correspondent déjà à des données Twitter. De plus, celui-ci intègre également un traitement des emojis en amont, lors de la tokenisation des tweets (la démojisation).

Cependant, comme nous avons pu le voir, appliquer de la démojisation lors du *fine-tuning* d'un modèle générique ne suffit pas à combler l'écart entre celui-ci et un modèle spécialisé. C'est également le cas lorsque le pré-apprentissage est poursuivi. Nous pouvons donc en déduire que la démojisation n'est pas concluante lorsqu'elle n'est pas effectuée lors du pré-apprentissage du modèle. Afin de pouvoir tout de même traiter les emojis, d'autres solutions sont donc à envisager, comme l'utilisation d'un *embedding* d'emojis, abordé en partie 3.3.4.3.

De plus, les spécificités de Twitter ne sont pas uniquement liées au vocabulaire employé. En effet, le texte contient également des subtilités sémantiques, et en particulier du sarcasme. Ce sarcasme peut être marqué par la présence d'un emojis en contradiction avec le sentiment du texte, ou tout simplement avec une opposition entre la situation décrite dans le tweet, et son sentiment. Prendre en compte ce sarcasme pourrait permettre de corriger un classifieur de sentiments en cas de tweet ironique.

Nous pensons donc qu'il est possible d'améliorer la classification de sentiments, d'une part, en appliquant un traitement des emojis supplémentaire à la démojisation, et d'autre part, en prenant en compte le sarcasme, très présent sur les réseaux sociaux. Ainsi, dans le chapitre suivant, nous proposons de combiner ces deux types d'informations avec le modèle BERTweet affiné sur de l'analyse de sentiments afin d'étudier l'impact que peuvent avoir la détection de sarcasme et l'utilisation d'une représentation d'emojis sur cette tâche.

Chapitre 4

Étude de l'impact de l'ajout de caractéristiques d'emojis et de sarcasme sur l'analyse de sentiments

Résumé

Dans ce chapitre, nous cherchons à améliorer la classification du sentiment. Pour cela, nous tentons d'ajouter des informations d'emojis et de sarcasme aux caractéristiques de BERT [33] à un modèle de classification de sentiments BERTweet [104] affiné sur cette tâche), en comparant différentes méthodes de combinaison (concaténation, somme et moyenne). Concernant les modes de combinaison, il en ressort que la somme et la moyenne (méthodes proches mathématiquement) permettaient d'obtenir les meilleurs résultats. Pour la combinaison d'information, cela permet de constater que l'ajout d'Emojional [8] améliore grandement la classification de sentiments. L'ajout d'informations de sarcasme permet également d'améliorer l'analyse de sentiments, cependant les résultats restent inférieures aux précédents. Enfin, pour l'ajout des deux types d'informations à la fois, les résultats, bien que supérieurs à la baseline, sont en dessous de ceux obtenus avec chaque élément indépendemment. Nos différents résultats nous ont permis d'émettre deux hypothèses : la première est que les modes de combinaison testés sont trop naïfs et ne permettent pas au modèle d'assimiler toutes les informations et la seconde concerne un éventuel conflit entre les informations apportées par les emojis et celles des caractéristiques de sarcasme. Ces deux hypothèses ne sont pas étudiées dans le cadre de cette thèse.

Sommaire

Étude de l'impact de l'ajout de caractéristiques d'emojis et de sarcasme sur l'analyse de sentiments

Le vocabulaire employé sur Twitter est très spécifique. En effet, la limitation de la longueur des messages à 280 caractères (140 caractères jusqu'à 2017) pousse les utilisateurs à utiliser des abréviations ou même à supprimer des mots. De plus, les messages sont structurés par la présence de mentions d'utilisateur, de hashtags ou encore d'emojis. Lors du traitement de tweets, les emojis sont rarement pris en compte [32]. Il est pourtant utile de les traiter, car ils apportent de l'information et peuvent permettre d'améliorer la classification de sentiments [87, 140] ou les systèmes de Questions-Réponses (*Question Answering*) [32].

Le sarcasme, très présent sur Twitter, est également important pour l'analyse de sentiments. (Parmar et al., 2018) définissent le sarcasme comme étant entre autres : un conflit entre une situation négative et un sentiment positif, ou l'inverse [114]. Il s'agit de l'une des difficultés pour la classification de sentiments, car s'il n'est pas détecté et pris en compte, il peut induire le classifieur en erreur. Dans l'exemple 5, l'emoji permet à un humain de déduire que le tweet est sarcastique, et donc que le sentiment est négatif. Pourtant, il sera considéré comme positif par un détecteur de sentiments qui analysera uniquement le texte, sans prendre en compte le sarcasme ou les emojis. C'est également le cas pour l'exemple 6, qui ne contient pas d'emoji, mais pour lequel un humain peut tout de même percevoir le sarcasme présent dans le texte. Nous avons donc pris le parti de traiter la détection de sarcasme en amont de la détection de sentiments, afin d'utiliser cette information comme unne méta-caractéristique dans le but d'améliorer la classification du sentiment [96].

Dans ce chapitre, nous proposons d'étudier l'impact de l'ajout d'information de sarcasme et d'emojis sur la détection de sentiments dans des tweets. Plus précisément, notre objectif est d'améliorer notre classifieur de sentiments, d'une part grâce à l'ajout de ces informations, et d'autre part en les intégrant de la meilleure manière, et ce, en réalisant également une étude sur les méthodes de combinaison de modèles. Pour cela, nous voyons dans un premier temps les emojis ainsi que leurs traitements, puis nous nous intéressons à la détection de sarcasme. Nous faisons ensuite un panorama des méthodes permettant de combiner ces deux informations, avant d'étudier l'impact que peut avoir la combinaison d'un *embedding* d'emojis et d'un classifieur de sarcasme sur la détection de sentiments.

Exemple 5 "What a wonderful day 😑"

Exemple 6 "Oh, I'm sorry. I didn't realize you're an expert on my life and how I should live it. Please continue while I take notes."

4.1 Qu'est ce qu'un emoji et comment peuvent-ils être traités ?

Dans cette partie, nous commençons par définir ce qu'est un emojis. Nous terminons ensuite par voir de quelles manières ils sont traités dans la littérature.

4.1.1 Définition d'un emoji

Les emojis ont été créés à la fin des années 1990 par Shigetaka Kurita. Ce sont des pictogrammes représentant un sentiment, une émotion, un objet ou même une idée. Le terme *emoji* est issu du japonais et signifie « caractère image ». Utilisable de la même manière que les émoticônes, ils sont cependant plus nombreux, et permettent donc d'exprimer plus de choses, comme des actions ou des lieux. Bien que les systèmes soient proches, il ne faut pas confondre les emojis avec les émoticônes ou encore les kaomojis, qui sont l'équivalent des émoticônes japonais. En effet, les émoticônes et kaomojis utilisent des caractères du clavier, tandis que l'emoji est une image utilisée en tant que caractère. Les kaomojis sont généralement plus complexes que les émoticônes, et sont surtout utilisés au Japon. Dans le tableau 4.1, nous pouvons voir un exemple d'émoticône, de kaomoji et d'emoji.

Les emojis ont fait leur première apparition avec l'opérateur téléphonique japonais NTT DoCoMo [1], puis ont été popularisés à travers le monde lorsque Apple [2] les ont intégrés à l'iPhone. Au départ, chaque logiciel avait sa propre représentation des emojis, de même que les codes auxquels ils étaient associés. Cependant, leur usage étant de plus en plus courant, les emojis les plus fréquemment utilisés ont été standardisés et sont maintenant supportés par Unicode depuis 2010.

Les emojis ont donc su se faire une place dans nos moyens de communication, y compris sur Twitter. En effet, nombreux sont les messages postés sur Twitter qui contiennent des emojis. Ils permettent non seulement d'exprimer l'humeur de l'utilisateur, mais également d'illustrer ses propos, ou tout simplement de s'exprimer de manière plus graphique. En apprentissage automatique, et en particulier pour la tâche d'analyse de sentiments sur Twitter en TAL, il est courant de s'intéresser aux emojis contenus dans les tweets afin de les analyser. Les emojis sont plus populaires que les émoticônes ou les kaomojis, et sont donc bien plus utilisés. Ceux-ci peuvent contenir des informations importantes pour la classification. Cependant, s'agissant d'images, ils ne peuvent pas être abordés de la même manière que les caractères du texte, et nécessitent donc un traitement particulier. L'objectif de cette thèse étant la détection de la polarité des opinions dans des tweets, nous nous intéressons aux informations que peuvent nous apporter les emojis, plus présents que les émoticônes ou les kaomojis dans les réseaux sociaux. Dans la partie suivante, nous voyons donc quelles sont les différentes méthodes utilisées pour le traitement des emojis.

1. https://www.nttdocomo.co.jp/english/
2. https://www.apple.com/

Émoticône	:)
Kaomoji	(⊙ ‿ ⊙)
Emoji	😄

TABLE 4.1 – Visage souriant en émoticône, kaomoji et emoji

4.1.2 Différentes manières de traiter les emojis dans la littérature

Comme nous l'avons vu précédemment, les emojis sont des images utilisées comme caractères. Ils doivent donc bénéficier d'un traitement spécifique les incluant ou non à la classification du texte. Dans cette partie, nous voyons quelles sont les différentes méthodes permettant de traiter les emojis.

Tout d'abord, la solution la plus simple consiste à supprimer les emojis du texte [138]. Ce faisant, il ne reste plus que le texte à traiter avec la méthode souhaitée, sans être gêné par la présence d'emojis. En revanche, en les supprimant, l'information contenue dans les emojis est également perdue.

Il est également possible de représenter les emojis par leur présence (*one hot encoding*), leur décompte ou leur fréquence [50]. Ce genre de méthodes, également utilisées pour le texte, permet de représenter un emoji par un score correspondant à la présence, la fréquence, le décompte, ou encore le TF.IDF (voir Chapitre 2 pour plus de détails).

Une troisième méthode permettant de traiter les emojis est de les remplacer par du texte. Plusieurs solutions sont alors possibles : la démojisation [102], et le remplacement par la description [140]. Dans un premier temps, la démojisation [102] est une méthode permettant de remplacer un emojis par son texte. Cela signifie qu'à la place de l'image, le texte contiendra la chaîne de caractère correspondant au code de l'emoji (par exemple :thumbsup: au lieu de 👍). De cette manière, la représentation du texte prend la main pour apporter également une représentation à l'emojis. La seconde méthode de remplacement permettant une meilleure représentation de l'emoji est l'utilisation de sa description [140]. Avec cette approche, similaire à la précédente, l'emoji est remplacé par un texte correspondant à sa description (« A thumbs-up gesture indicating approval. », au lieu de 👍). Encore une fois, cela permet au système de représentation du texte de trouver également une représentation pour l'emoji.

Pour terminer, il est possible de représenter les emojis avec un vecteur appris afin d'être traité par un classifieur. En effet, les emojis peuvent également être projetés dans un espace latent qui fournit une représentation commune avec le texte grâce à l'utilisation d'un *embedding* unique. Trois solutions sont alors possibles. Dans un premier temps, chaque emoji peut être présent dans l'espace et donc correspondre à un *token*[3]. Chaque emoji aura donc sa propre représentation [6, 32, 140]. Cela est possible à condition d'inclure les emojis dès le pré-apprentissage de la représentation

3. Instance d'une séquence de caractères dans un document, regroupés comme une unité sémantique utile pour le traitement Stanford NLP Group

vectorielle. Il est également possible de remplacer les emojis dans le texte par un tag unique. C'est ensuite ce tag qui sera présent dans l'espace. Cela permet d'avoir une représentation unique pour tous les emojis. Enfin, il est également possible de les traiter grâce à l'utilisation d'un espace de représentation spécifique aux emojis, combiné à une représentation pour le texte. Cette méthode est proche de la précédente, cependant ici les emojis ont leur propre espace de représentation. Plusieurs solutions existent pour cet espace de représentation. Tout d'abord, (Chen et al., 2018) ont construit un système de représentation fondé sur VADER [4] [60], permettant d'associer deux vecteurs à chaque emoji : un vecteur positif, ainsi qu'un vecteur négatif [21]. VADER est un outil d'analyse de sentiments fondé sur des règles et du lexique spécialisé dans les sentiments exprimés dans les réseaux sociaux. Grâce à celui-ci, les données collectées sont annotées en sentiment positif ou négatif. Puis, pour chaque emoji, la représentation est apprise : le vecteur positif est entraîné sur les tweets annotés positivement dans le jeu de données, et le vecteur négatif sur les données annotées négativement. Les deux représentations suivantes sont similaires car apprises avec les même méthodes : Emoji2vec [35] et Emojional [8]. La première, Emoji2vec est une adaptation sur des emojis de l'algorithme d'*embedding* Word2vec, fondé sur le skip-gram [99] de Google. Les données Twitter collectées pour l'apprentissage de cette représentation a permis d'obtenir 6 088 descriptions de 1 661 emojis. Emojional [8] quant à lui est fondé sur Emoji2vec, utilisant donc également l'algorithme du skip-gram. Cette fois, cela correspond à 10 854 descriptions, permettant de représenter 1 816 emojis.

Maintenant que nous avons vu les différents traitements des emojis, la section suivante permet de discuter de la pertinence de chacune de ces approches pour notre problématique.

4.1.3 Discussion

Nous avons pu voir ce qu'était un emoji et de quelles manières ils pouvaient être traités. Parmi les traitements cités dans la partie précédente, la suppression des emojis peut rapidement être éliminée. En effet, bien que cette méthode soit la plus simple à appliquer, supprimer les emojis constitue une perte d'information importante [65].

Concernant la démojisation [5], nous avons pu l'expérimenter dans le Chapitre 3, il s'agit donc de notre *baseline* concernant les emojis. De plus, celle-ci est déjà appliquée par BERTweet, le modèle utilisé pour nos expérimentations. Ces informations sur les emojis sont donc déjà contenues dans notre classifieur de sentiments. Remplacer les emojis par leur description est semblable au principe de la démojisation. Cette approche a donc été écartée.

Afin de prendre les emojis en compte, nous nous appuyons sur l'utilisation d'une représentation vectorielle d'emojis, et nous avons sélectionné Emojional[8] car elle permet de représenter un plus grand nombre de ces emojis. Tout comme pour le texte, les méthodes d'*embedding* permettent une meilleure représentation des emojis qu'avec leur

4. https://github.com/cjhutto/vaderSentiment
5. Remplacement par le texte correspondant

présence/absence. La représentation vectorielle sera donc combinée à notre classifieur de sentiments, BERTweet [104], qui inclut également une étape de démojisation lors de la tokenisation des tweets. Ceci en fait donc un modèle final hybride, dans lequel les emojis sont représentés à deux niveaux différents : dans le texte après démojisation, et combiné à la représentation du texte.

4.2 Qu'en est-il du sarcasme ?

Les emojis font partie du texte, il suffit donc de les extraire pour pouvoir leur appliquer un pré-traitement, ou les considérer comme une caractéristique à ajouter à un modèle. Dans le cas du sarcasme, il s'agit en revanche d'une information de plus haut niveau, puisqu'elle est contenu dans la sémantique des tweets. Les méthodes de détection de sarcasme sont donc plus complexes que celles de traitement des emojis. Puisque nous souhaitons analyser l'impact de l'incorporation de caractéristiques d'emojis et de sarcasme à un classifieur de sentiments, nous étudions dans cette partie ce qu'est le sarcasme et comment le détecter, en sélectionnant les méthodes les plus adaptées à notre problématique.

4.2.1 Définition

La détection de sarcasme est une branche de l'analyse de sentiments en TAL. Cette tâche est à ne pas confondre avec la détection d'ironie, en effet l'ironie est un type de sarcasme, la détection d'ironie est donc une sous-tâche de la détection de sarcasme.

La détection de sarcasme est une tâche compliquée, car il n'y a pas de réelle frontière entre du sarcasme et du non sarcasme. La perception du sarcasme varie d'une personne à l'autre, et dépend de la culture, du contexte, etc. À l'écrit, la détection de sarcasme est encore plus difficile : en lisant un message sur les réseaux sociaux, nous n'avons pas accès à l'intonation du locuteur, ou l'expression de son visage. Les seuls indices accessibles à un modèle sont donc l'utilisation d'un hashtag (comme « #sarcasm » ou « #no »), la ponctuation, ainsi que les emojis présents dans le tweet.

Correctement détecter le sarcasme peut améliorer l'analyse de sentiments. Comme ont pu le démontrer (Maynard and Greenwood, 2014), la détection de sarcasme, qu'ils ont associé à des règles définissant la portée du sarcasme, permet d'améliorer la détection de sentiments dans des tweets [96]. En effet, le sarcasme peut également être une information importante à prendre en compte, car elle peut perturber la détection de sentiments ou l'induire en erreur. Par exemple, dans un tweet sarcastique, la polarité du sentiment peut être inversée. Dans l'exemple que nous avons vu en introduction, "What a wonderful day 😑", la présence de l'emoji « 😑 » change la polarité de positive vers négative, puisque l'on comprend que l'auteur de ce tweet est sarcastique.

Inversement, détecter le sentiment permet de mieux détecter le sarcasme. (Majumder et al., 2019) ont mis au point un réseau multi-tâche permettant de représenter d'une part le sentiment, et d'autre part le sarcasme [92]. Les deux représentations sont ensuite combinées, ce qui permet au classifieur final de prédire le sarcasme.

4.2.2 Détection de sarcasme fondée sur le contexte

S'il existe plusieurs sous-tâches à la détection de sarcasme, deux approches permettent d'effectuer cette tâche de classification. Dans cette partie, nous décrivons la première approche : la détection de sarcasme fondée sur le contexte.

Pour ces méthodes, le contenu du tweet est considéré comme insuffisant. Le contexte du message est donc utilisé afin d'apporter des informations supplémentaires. Par exemple, ils peuvent être utilisés comme contexte les réponses aux tweets, ou encore l'historique des messages auquel il appartient. En effet, (Wallace et al., 2014) ont démontré que les modèles classiques avaient du mal à classifier le sarcasme lorsque du contexte était nécessaire aux humains pour le détecter [156]. Pour pallier le manque de contexte (Poria et al., 2016) utilisent des informations supplémentaires comme le sentiment, les émotions et la personnalité pour classifier un texte [117]. (Khattri et al., 2015), (Rajadesingan et al., 2015), (Zhang et al., et 2016) quant à eux, se servent de l'historique des messages publiés par les utilisateurs comme contexte [71, 123, 172]. Enfin (Wallace et al., 2015) exploitent un jeu de données Reddit [6] de sarcasme [156] afin d'extraire des caractéristiques de contexte [157]. Ceci leur a permis d'améliorer la classification de sarcasme verbal. Ces approches fondées sur le contexte nécessitent d'avoir accès aux historiques, ou plus globalement au contexte de publication des messages. Mis à part une information de retweet présente dans les tweets, et permettant d'indiquer qu'un tweet a été partagé, ces données de contexte ne sont pas toujours facilement accessibles. Notre tâche principale étant l'analyse de sentiments, aucun contexte n'a été collecté lors de la constitution des jeux de données contenus dans notre ensemble de corpus. Cependant, puisque nous cherchons à améliorer la classification du sentiment en ajoutant des informations de sarcasme, nous nous intéressons à la détection de sarcasme fondée sur le contenu.

4.2.3 Détection de sarcasme fondée sur le contenu

Après avoir vu les méthodes fondées sur le contexte, nous présentons plus en détail la détection de sarcasme fondée sur le contenu. Ces méthodes se concentrent sur le contenu du texte pour classifier un message selon le vocabulaire utilisé, ou sur des indicateurs présents dans le tweet. Par exemple (Carvalho et al., 2009) se sont servi de caractéristiques linguistiques telles que les prédicats positifs, les interjections, ainsi que des indices oraux et gestuels comme les émoticones, les onomatopées (rires), l'utilisation de nombreux signes de ponctuation, etc. [19]. (Tepperman et al., 2006) ont étudié la détection de sarcasme dans le discours oral en utilisant la prosodie et les spectrogramme [147]. Nous pouvons aussi citer (Davidov et al., 2010) et (Tsur et al., 2010) qui ont fondé leur classifieur sur des *patterns* syntaxiques [31, 149]. (Joshi et al., 2015) quant à eux, se sont appuyé sur une utilisation de caractéristiques lexicales, pragmatiques, etc [67]. Il est également possible de considérer le sarcasme comme l'association d'un sentiment positif avec une situation négative, comme l'ont fait (Riloff et al., 2013) [125].

Notre objectif est de pouvoir améliorer les performances de notre classifieur de sen-

6. https://www.reddit.com/

timents. Nous avons donc cherché dans la littérature du sarcasme, et en particulier dans les méthodes fondées sur le contenu, des outils permettant d'améliorer cette tâche. Nous pouvons diviser les types d'algorithmes utilisés en trois catégories : les approches fondées sur des règles, les approches fondées sur des caractéristiques et enfin les approches avec apprentissage. Dans cette partie, nous étudions ces trois types d'approches.

Méthodes fondées sur des règles

Le but de la détection de sarcasme fondée sur des règles est d'identifier la présence de sarcasme à travers la mise en place de règles créées manuellement ou apprises. Ces règles permettent de mettre en évidence certains indicateurs de sarcasme dans le texte. (Veale and Hao, 2010) ont étudié la présence de sarcasme dans les comparaisons (de la forme « [...] as a [...] ») [152]. (Maynard and Greenwood, 2014) se focalisent sur les hashtags présents dans les tweets, en vérifiant si le sentiment présents dans un hashtag entre en contradiction avec le contenu du tweet [96]. (Bharti et al., 2015) ont créé un classifieur fondé sur deux règles [13]. Dans un premier temps, les arbres syntaxiques de chaque phrase sont générés, permettant ainsi d'identifier les passages contenant un sentiment, et plus particulièrement de détecter si une phrase positive contient des expressions négatives. Dans un second temps, la deuxième règle a pour objectif de détecter les hyperboles. Enfin, (Riloff et al., 2013) détectent le sarcasme en recherchant des verbes positifs employés dans des situations négatives [125].

Méthodes fondées sur des caractéristiques

La plupart des approches décrites dans cette partie utilisent les « sacs de mots » comme caractéristique principale. (Tsur et al., 2010) mettent en évidence la présence de motifs discriminants extraits d'un grand corpus annoté en sarcasme [149]. Comme décrit précédemment, (Riloff et al., 2013) ont créé un classifieur de sarcasme hybride : d'une part, ils tentent de détecter un contraste entre la polarité du texte et la situation (texte négatif pour une situation positive, ou inversement), et d'autre part ils utilisent un classifieur de sarcasme [125]. Le tweet sera donc considéré comme sarcastique s'il est identifié soit par le classifieur, soit par le détecteur de contraste. Des caractéristiques lexicales peuvent également être employées [5, 49]. (Reyes et al., 2012) fondent leur approches sur la recherche d'ambiguïté [124]. (Liebrecht et al., 2013) quant à eux recherchent les marqueurs d'émotions, en utilisant également les n-grams comme caractéristiques [83]. Pour (Hernández-Farías et al., 2015), c'est la similarité sémantique qui leur permet de détecter le sarcasme, ainsi que des caractéristiques liées à la longueur du texte ou à la capitalisation [55]. (Subramanian et al., 2019) se basent sur l'utilisation des emojis pour détecter le sarcasme [143]. D'autres approches telles que (Barbieri and Saggion, 2014) utilisent une combinaison de caractéristiques définies manuellement telles que la fréquence, le style de langage (parlé/écrit), la structure du texte, le sentiment, etc [4]. Enfin, des caractéristiques linguistiques [19] peuvent également servir à la détection du sarcasme.

Méthodes supervisées

Parmi les approches supervisées, nous pouvons distinguer deux sous-catégories : d'une part les approches traditionnelles d'apprentissage automatique, et d'autre part, les approches fondées sur les réseaux de neurones (voir Chapitre 2 pour plus de détails sur les

algorithmes supervisés). Dans un premier temps, la grande majorité des approches traditionnelles d'apprentissage automatique utilisent des SVM [31, 67, 147, 149]. (Reyes et al., 2012) ont également utilisé Naives Bayes, en plus du SVM pour la détection de sarcasme [124]. (Riloff et al., 2013) ont pu comparer les SVM avec les approches fondées sur des règles citées précédemment [125]. En ce qui concerne les approches utilisant les réseaux de neurones, leur utilisation s'est de plus en plus répandue ces dernières années pour la détection de sarcasme. (Joshi et al., 2016) ont utilisé la distance entre les représentations vectorielles des mots (*word-embeddings*) comme caractéristiques, qu'ils ont augmenté de quatre autres caractéristiques fondées sur les similarités [68]. (Ghosh and Veale, 2016) comparent des SVM récursifs à des réseaux de neurones [45]. Plus particulièrement, ils réalisent différentes combinaison de couches convolutives avec des couches de LSTM et des couches linéaires, montrant ainsi que les méthodes fondées sur l'apprentissage profond atteignent de meilleures performances. Le modèle CASCADE [52] propose une méthode hybride utilisant à la fois le contenu et le contexte. En effet, le contenu est modélisé grâce à un CNN, tandis que le contexte est traité grâce à une représentation vectorielle d'utilisateurs (*user embedding*), ainsi que des caractéristiques du discours de tous les utilisateurs et messages d'un forum. Enfin, des méthodes plus récentes, fondées sur les Transformer ont également fait leur preuves sur la détection de sarcasme.

4.2.4 Discussion

Dans notre cas, nous nous intéressons à l'analyse de sentiments, et plus particulièrement à la classification de la polarité du sentiment. Par conséquent, nous chercherons à améliorer cette classification en utilisant la détection de sarcasme, et non l'inverse. Les approches utilisant l'analyse de sentiments pour l'amélioration de la détection de sarcasme ne correspondent donc pas à notre tâche, même s'il est possible de s'en inspirer.

Concernant le type de détection de sarcasme, nous nous concentrons uniquement sur les méthodes fondées sur le contenu. En effet, puisque notre objectif principal reste la classification de sentiments, les corpus publics qui correspondent à cette tâche ne contiennent pas d'informations de contexte. Ainsi, même les tags de retweet sont supprimés lors du nettoyage des données effectué après la collecte des tweets.

Les approches fondées sur des règles sont aujourd'hui rarement utilisées puisqu'elles sont limitées aux cas couverts par les règles. De plus, le sarcasme est une information sémantique, et les approches fondées sur des règles ne permettent pas de modéliser la sémantique du texte. Concernant les méthodes fondées sur des caractéristiques, elles reposent principalement sur les méthodes « sac de mots » (*Bag-of-words*, ou BOW) combinées à d'autres caractéristiques créées manuellement, comme des listes de vocabulaires, des n-grammes, ou encore des caractéristiques syntaxiques, morphologiques ou même structurelles dans le texte. Bien que ce genre de méthodes permette d'intégrer différentes informations pouvant être utiles à la classification, la représentation du texte en « sac de mots » n'est quasiment plus utilisée, au profit des représentations vectorielles avec apprentissage (*embedding)*, qui permettent une meilleure représentation de la sémantique du mot (deux mots proches sémantiquement seront proches dans l'es-

pace de représentation) [99]. De plus, le choix des caractéristiques utilisées est décidé manuellement et peut ne pas être optimale, ou peut même biaiser la prédiction.

Les modèles Transformer font partie des méthodes avec apprentissage et sont actuellement à l'état de l'art. Les méthodes basées sur ces modèles ont l'avantage d'apprendre des caractéristiques sémantiques et syntaxiques lors de leur pré-entraînement [24, 27, 48, 58, 63, 85, 155]. Puisque le sarcasme est une information sémantique, nous nous sommes naturellement penchés vers ces méthodes pour le traiter.

Tout comme (Savini and Caragea, 2022), nous avons choisi d'utiliser un modèle Transformer [151] actuellement à l'état de l'art [141], le modèle BERT [33]. Ce modèle est affiné, ce qui permet d'adapter ce modèle générique à la classification de sarcasme. En effet, affiner le modèle permet de palier le manque de données annotées en grandes quantités. Cependant, plutôt que d'utiliser BERT afin de classifier les tweets, nous utilisons cette information de sarcasme en tant que caractéristique, combiné à d'autres représentations, à l'instar de [96] qui utilisent la détection de sarcasme comme simple caractéristique pour inverser ou non la polarité du sentiment.

L'information de sarcasme étant modélisée avec BERT, nous obtenons donc un espace latent de 768 dimensions, qu'il nous faut maintenant combiner au modèle d'analyse de sentiments BERTweet, avec 768 dimensions également, et avec Emojional, avec 300 dimensions. La partie suivante concerne ces méthodes de fusion.

4.3 De quelle manière pouvons-nous combiner ces éléments ?

Afin d'améliorer la classification de sentiments, ou encore de traiter plusieurs types de données à la fois, il peut être intéressant de combiner différents modèles ou plus simplement différentes représentations. Dans notre cas, nous souhaitons étudier l'impact que peut avoir l'ajout de caractéristiques d'emojis et d'un modèle de classification de sarcasme sur l'analyse de sentiments. En effet, si les emojis sont présents dans le texte et peuvent être facilement extraits, le sarcasme est une information sémantique de haut niveau. Il faut donc être prudent sur la méthode de fusion de ces informations.

Pour cela, nous nous intéressons donc au multi-modèle (ou fusion/combinaison de caractéristiques). Dans cette partie, nous faisons donc un panorama sur la fusion de modèle. Nous voyons dans un premier temps quels sont les éléments généralement associés, puis nous nous intéressons à la façon dont ces éléments sont combinés entre eux. Pour finir, nous avons une discussion sur ces différentes méthodes. Ceci nous permet de déterminer quels sont les modes de fusion les plus adaptés afin de combiner un *embedding* d'emojis et un modèle de détection de sarcasme, qui sont deux types de caractéristiques différentes, à un modèle d'analyse de sentiments.

4.3.1 Quels éléments sont combinés ?

Si fusionner différentes caractéristiques permet d'apporter des informations supplémentaires à un modèle, encore faut-il savoir quels modèles ou représentations combi-

ner. Cette partie nous permet d'identifier les représentations ou modèles couramment fusionnés dans la littérature.

Dans le cas de la classification multi-modale, ce sont des caractéristiques de nature différente qui sont combinées. (Chordia and BG, 2020) combinent des caractéristiques de texte et d'image [23]. En effet, ils fusionnent CamemBERT [93], le modèle BERT entraîné sur des données en français, avec un ResNet [53], un réseau de neurones pour le traitement d'image. Ils tentent également la combinaison de différentes représentations pour le texte, parmi CamemBERT, [93], FlauBERT [78], qui est un autre BERT appris sur du français, et M-BERT, un BERT entraîné sur des données multilingues [33].

Parmi les éléments les plus combinés, nous retrouvons les représentations vectorielles. Il est courant de mélanger différentes représentations de textes ou d'images afin d'en obtenir une meilleure. Par exemple, (Zhang et al., 2020) ont combiné le *word-embedding* GloVe à la représentation de BERT [175]. Dans le cas de (Ghosh and Veale, 2016), ce sont quatre représentations différentes qui sont utilisées pour la classification d'articles scientifiques : RoBERTa, RoBERTa + LDA [7], RoBERTa au niveau des phrases, et enfin, un TF.IDF suivis d'une régression logistique (voir Chapitre 2 pour plus d'information sur ces approches). (Wang et al., 2020) proposent *Automated Concatenation of Embeddings* qui permet de trouver la meilleure concaténation d'*embedding*. Enfin, des représentations d'éléments différents du texte, comme les emojis, peuvent être fusionnées aux représentations textuelles. Par exemple, (Islam and Goldwasser, 2020) utilisent Emoji2vec en complément de Word2vec pour la représentation de tweets [61]. Les deux *embeddings* sont fusionnés grâce à un bi-LSTM et à de l'attention, puis cette combinaison est concaténé à des meta-informations issues du profil de l'utilisateur Twitter.

Enfin, une méthode plus simple permet également d'intégrer différentes informations en tant que caractéristique. En effet, nous pouvons voir que (Maynard and Greenwood, 2014) détectent le sarcasme grâce au contenu des hashtags [96]. La polarité du sentiment est ensuite inversée en fonction des hashtags et de règles définies manuellement.

Après avoir vu quels éléments fusionner, la partie suivante est consacrée aux méthodes de combinaison.

4.3.2 De quelle manière sont-ils combinées ?

Une fois que les types d'éléments à combiner ont été identifiés, il reste à définir si et comment ils vont être fusionnés ensemble. En effet, certaines méthodes de combinaison ne sont applicables que dans certaines configurations précises. Dans cette partie, nous nous intéressons donc aux différentes méthodes permettant de faire du multi-modèle.

La stratégie la plus souvent appliquée dans les ensembles de modèles est le vote à la

7. Modèle probabiliste génératif qui permet de décrire des collections de documents de texte ou d'autres types de données discrètes. [15]

majorité. Cette méthode de fusion est utilisable uniquement lorsque plusieurs classifieurs correspondant à la tâche sont en jeu. Par exemple, (Ghosh and Veale, 2016) réalisent un ensemble de quatre modèles afin de classifier des articles scientifiques, et la décision finale est prise en utilisant la classe votée majoritairement par les quatre modèles [45]. C'est également le cas pour (Zhang et al., 2020) qui se servent aussi du vote à la majorité pour leur trois classifieurs afin d'identifier le discours haineux et offensif [175]. Le vote à la majorité peut également être pondéré [69].

(Lekshmiammal and Madasamy, 2021) ont choisit une stratégie différente pour leur ensemble, puisqu'ils réalisent leur classification de *Fake News* en choisissant comme prédiction finale la somme des probabilités prédites par chaque modèle, la classe de sortie étant l'*argmax* de ces probabilités [81]. Moyenner les prédictions est également possible [21]. Une fois de plus, il est nécessaire d'avoir plusieurs classifieurs pour pouvoir utiliser cette méthode. Ce type d'approche n'est donc pas adapté à la combinaison de représentation.

(Chordia and BG, 2020) ont voulu combiner des caractéristiques de texte et d'image [23]. Pour cela, ils comparent l'utilisation de la concaténation, de la transformation bi-linéaire [8] et du produit scalaire à la co-attention [90]. (Dai et al., 2021) proposent une approche qui se veut universelle fondée sur l'attention, qu'ils comparent à d'autres approches, telles que la somme, ou la concaténation [28]. Pour ces méthodes de combinaison, il n'est pas nécessaire d'utiliser plusieurs classifieur. En effet, les différentes informations sont combinées en amont de la prise de décision. Cela permet donc de fusionner différentes caractéristiques, comme différentes représentation du texte, ou même des informations de natures différentes. En ce qui concerne le cas particulier du sarcasme avec le sentiment, (Majumder et al., 2019) ont pu fusionner ces deux types d'information en réalisant un apprentissage multi-tâche [92]. En effet, ils tentent de classifier à la fois le sentiment et le sarcasme. Les classifications sont effectuées simultanément, avec la particularité que la représentation de sentiments sert également à la classification de sarcasme, les deux représentations étant combinées par le biais d'une couche de *Neural Tensor Network* (voir figure 4.1). Il s'agit d'une couche bi-linéaire permettant la fusion de deux Tenseurs, et utilisant notamment de la concaténation.

(Kiela et al., 2018) ont mis en place une méthode différente de celles décrites précédemment appelée *meta-embedding* dynamique. Celle-ci permet d'utiliser plusieurs représentations différentes, et laisse le modèle sélectionner la représentation la plus adaptée grâce à un apprentissage. Cela permet par exemple d'utiliser plusieurs *embeddings* à la fois, comme GloVe [115], Word2vec [99] et FastText [16] sans avoir à gérer l'utilisation de chacun d'entre eux.

4.3.3 Discussions

Nous venons de voir différentes méthodes permettant la combinaison de caractéristiques. Nous rappelons que nous souhaitons ajouter des caractéristiques d'emojis et de

8. Méthode de traitement numérique du signal, permet de transformer les représentations du système en temps continu en temps discret (et inversement).

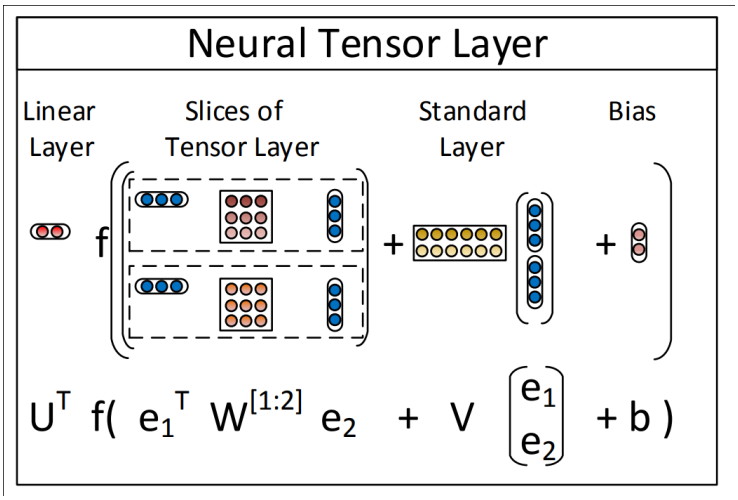

FIGURE 4.1 – *Neural Tensor Network* permettant la combinaison de deux Tenseurs

sarcasme à BERTweet [104], notre classifieur de sentiments, et étudier l'impact de ces ajouts. En effet, nous souhaitons que notre modèle puisse apprendre automatiquement comment utiliser ces différentes informations pour effectuer de l'analyse de sentiments, et non nous contenter d'une inversion de polarité du sentiment en cas de détection de sarcasme [96]. Les éléments à combiner ainsi que leurs représentations sont donc déjà définis, il nous faut maintenant déterminer comment fusionner ces caractéristiques de manière simple, mais efficace. Il s'agit de trois représentations ayant des informations différentes, et donc des espaces de projections différents. Ces espaces doivent être rapprochés afin que les trois types d'informations puissent être combinés de la meilleure manière possible.

Les méthodes de votes à la majorité sont incompatibles avec notre approche. En effet, comme dit précédemment, seuls un classifieur peut voter, et ceux-ci doivent être adaptés à la tâche. Ainsi, faire voter une représentation d'emojis et un classifieur de sarcasme dans le cadre d'analyse de sentiments n'aurait pas de sens. De la même manière, sommer les probabilités avant de déterminer la classe de sortie ne serait possible qu'en réalisant une classification avec chacune des caractéristiques combinées. La transformation linéaire appliquée par (Chordia and BG, 2020) [23] étant spécifique au type de données utilisées, celle-ci n'est pas compatible avec notre approche.

Nous souhaitons combiner trois informations différentes, et non sélectionner la représentation la plus adaptée pour chacun de nos tweets. Le *meta-embedding* dynamique n'est donc pas adapté à notre problématique, ainsi, nous n'emploierons pas cette approche pour notre fusion de modèles.

Contrairement aux solutions décrites précédemment qui tentent d'intégrer le sarcasme en utilisant des règles et des *a priori* (contraste entre la polarité du texte et la situation, ambiguïté, etc.), nous pensons qu'il est plus judicieux de laisser le modèle apprendre comment prendre en compte ces informations afin de ne pas introduire d'erreur. De cette manière, nous espérons que le modèle apprenne à faire les bons choix au bon moment.

Nous utilisons donc la somme, la concaténation, ainsi que la moyenne, qui sont majoritairement utilisées dans la littérature [23, 28, 92, 160]. Ces méthodes ont également l'avantage d'être rapidement modifiables pour effectuer nos expérimentations. Maintenant que nous avons clarifié les méthodes de combinaison de caractéristiques, et que les états de l'art sur les emojis et la détection de sarcasme ont été effectués, nous décrivons dans la section suivante nos différentes expérimentations.

4.4 Est-il possible d'améliorer l'analyse de sentiments en donnant des informations supplémentaires au classifieur ?

L'étude de ces différents articles montre que le traitement des emojis et la détection de sarcasme peuvent être utiles à l'analyse de sentiments. Dans cette section, après avoir vu le protocole expérimental, nous commençons par combiner une représentation d'emojis, et plus précisément Emojional, à BERTweet afin d'améliorer la classification de sentiments. Puis, en nous inspirant de (Majumder et al., 2019) et de (Maynard and Greenwood, 2014), nous proposons d'intégrer l'information de sarcasme en amont de la détection de sentiments [92, 96]. Cependant, plutôt que d'inverser automatiquement la polarité du sentiment en fonction de la détection de sarcasme [96], ou encore d'utiliser le sentiment pour mieux détecter le sarcasme [92], nous présentons un modèle capable d'apprendre automatiquement comment utiliser cette information de sarcasme pour la détection de sentiments.

Nous proposons de tester deux approches. Dans un premier temps, nous combinons seulement la représentation d'emojis au classifieur de sentiments, et dans un second temps, nous proposons de fusionner à la fois l'information de d'emojis et de sarcasme avec le texte, ce qui, à notre connaissance, n'a pas été proposé dans la littérature. Ceci nous permettra d'étudier l'impact de l'utilisation de ces deux caractéristiques différentes sur l'analyse de sentiments. En effet, si chacun de ces éléments individuellement, combiné au traitement classique du texte, permet d'améliorer la détection de sentiments, alors le fait d'utiliser à la fois les emojis et le sarcasme en plus du texte devrait grandement augmenter l'efficacité de la classification de sentiments.

4.4.1 Protocole Expérimental

Cette partie est consacrée à description de notre protocole expérimental. Nous présentons chacun de ces éléments du protocole, et nous terminons par détailler les conditions d'expérimentation.

Baseline La *baseline* utilisée pour comparer nos expérimentations correspond aux résultats obtenus dans le Chapitre 3 en utilisant BERTweet. Ce modèle a été utilisé dans les conditions suivantes :

— modèle affiné sur chacun des jeux de données

— normalisation du texte par le Tokenizer (avec démojisation)

— application des mesures discutées en Chapitre 3 pour éviter l'oubli catastrophique, à savoir
  — gel des couches, sauf le dernier quart
  — taux d'apprentissage faible ($1e-5$)

Analyse de sentiments Pour notre étude, nous avons tout d'abord dû sélectionner le cœur de notre classifieur final : le modèle d'analyse de sentiments. Grâce à notre Chapitre 3, nous avons pu mettre en place des lignes directrices, permettant la classification de tweets en ayant accès à des ressources limitées. La quantité de ressources à notre disposition n'ayant pas changé, nous appliquons ces lignes directrices pour l'apprentissage de notre modèle d'analyse, à savoir :

— l'utilisation du *fine-tuning*, étant données la taille de nos jeux de données,
— l'utilisation d'un faible taux d'apprentissage, et du gel des couches du modèle (toutes excepté le dernier quart),
— utilisation d'un modèle spécialisé, ici BERTweet [104] pré-appris sur des données Twitter.

Nous avons donc utilisé l'espace latent du modèle BERTweet comme représentation du sentiment. Le modèle a été affiné sur notre jeu de données, avant d'être sauvegardé (poids de la dernière couche seulement) et importé dans le modèle de classification final.

Emojis Afin de pouvoir étudier l'impact de l'ajout des emojis, nous avons choisi d'utiliser une représentation vectorielle des emojis (voir section 4.1.3), et plus précisément Emojional [8]. Cette représentation est donc combinée à notre modèle d'analyse de sentiments, BERTweet, où un vecteur représentant le texte et un autre représentant les emojis seront fusionnés. Afin de normaliser la taille du vecteur d'emojis, le nombre maximum d'emojis contenu dans un tweet a été défini pour chaque jeu de données. Ceci nous permet d'obtenir une taille de vecteur fixe, en réalisant un *padding* lorsque le maximum n'est pas atteint. Pour ce *padding*, un vecteur « emoji vide » a été ajouté à Emojional. Afin que ce nouveau *token* puisse s'adapter à nos données, deux options ont été testées pour le modèle final : laisser la couche d'*embedding* gelée (comportement par défaut dans le modèle), ou la dégeler manuellement afin que les poids, et en particulier le vecteur représentant l'absence d'emoji, puissent être mis à jour.

Détection de sarcasme Concernant la détection de sarcasme, deux approches ont été testées. Dans un premier temps, nous avons appliqué une approche naïve : les corpus d'analyse de sentiments ont été annotés en sarcasme (deux classifieurs ont été comparés : BERT le T5), ce qui a permis d'ajouter un tag à la fin du texte pour les tweets sarcastiques. Pour la seconde approches, nous avons sélectionné un modèle de détection de sarcasme qui puisse également être combiné à notre modèle d'analyse de sentiments. Le modèle choisi est un BERT-base affiné sur un jeu de données de sarcasme : SemEval 2018 task 3 - irony detection, que nous appellerons BERT sarcasme. Les poids de la dernière couche du modèle ont ensuite été sauvegardés, avant d'être chargés dans le modèle final. Nous utilisons donc uniquement l'espace latent de ce modèle.

Combinaison Afin de fusionner les différents éléments, nous utilisons les trois méthodes de combinaison les plus courantes décrites section 4.3, à savoir la concaténation, la somme et la moyenne. Pour la concaténation, les poids de chacun des éléments sont chargés, concaténés en un grand vecteur, puis des couches Dense permettent d'arriver à la classification finale. Pour la somme et la moyenne, une étape de réduction de dimension doit être ajoutée afin que les modèles BERT (768 dimensions) et l'*embedding* d'emojis (300 dimensions) aient le même nombre de dimensions (voir figure 4.2 et 4.3).

Jeux de données Parmi les six jeux de données présentés dans le Chapitre 3, seulement quatre contiennent des emojis et par conséquent sont utilisables pour ces expérimentations : HatEval [10], OffensEval [170], IberEval [36]et SemEval 2018 task 3a [150]. Sur ces quatre jeux de données, SemEval 2018 task 3a a été utilisé pour le *fine-tuning* du modèle BERT dédié à la détection de sarcasme. Les trois modèles restants (HatEval, OffensEval et IberEval) ont donc été exploités pour nos expérimentations, c'est-à-dire dans un premier temps pour l'affinage du modèle d'analyse de sentiments, puis pour le modèle final. Malgré son manque d'emojis, le corpus de SemEval 2016 nous a tout de même servi pour tester l'intégration naïve du sarcasme.

Expérimentations Chaque expérimentation a été réalisée avec dix découpages aléatoires stratifiés, permettant ainsi de vérifier la généralisation des modèles. Il y a ensuite eu vingt epochs de *fine-tuning* par découpage, sans arrêt prématuré. Ce sont ensuite les performances de la meilleure epoch qui sont sauvegardées.

4.4.2 Inclure la représentation d'emojis dans le classifieur de sentiments

Comme nous avons pu le voir dans la littérature, la prise en compte des emojis en plus du texte pour l'analyse de sentiments fonctionne bien. Notre objectif étant d'étudier l'impact de l'ajout d'informations d'emojis sur l'analyse de sentiments, nous avons testé dans un premier temps de combiner le texte avec une représentation vectorielle des emojis, et plus précisément avec Emojional [8].

Le tableau 4.2 présente les résultats de nos premières expérimentations concernant l'utilisation de BERTweet pour traiter le texte des tweets, combiné à Emojional pour représenter les emojis. Ces résultats sont obtenus en utilisant la concaténation, la somme et la moyenne comme méthode de combinaison. De plus, chacune de ces méthodes est réalisée avec la couche d'*embedding* d'emojis gelée et dégelée, comme décrit précédemment.

Nous pouvons voir que cette combinaison obtient de meilleures performances que la *baseline* (jusqu'à +6,66 points de score F1 sur IberEval, +6,15 pour OffensEval et +5,19 pour HatEval). Nous pouvons donc en déduire que ce traitement des emojis apporte des informations non contenues dans le modèle BERTweet. Les performances sont particulièrement bonnes lorsque la couche d'*embedding* est dégelée. Notre hypothèse est que la représentation des emojis s'adapte à nos données ainsi qu'à notre tâche, et que les poids du *token* ajouté, représentant l'absence d'emojis, sont appris.

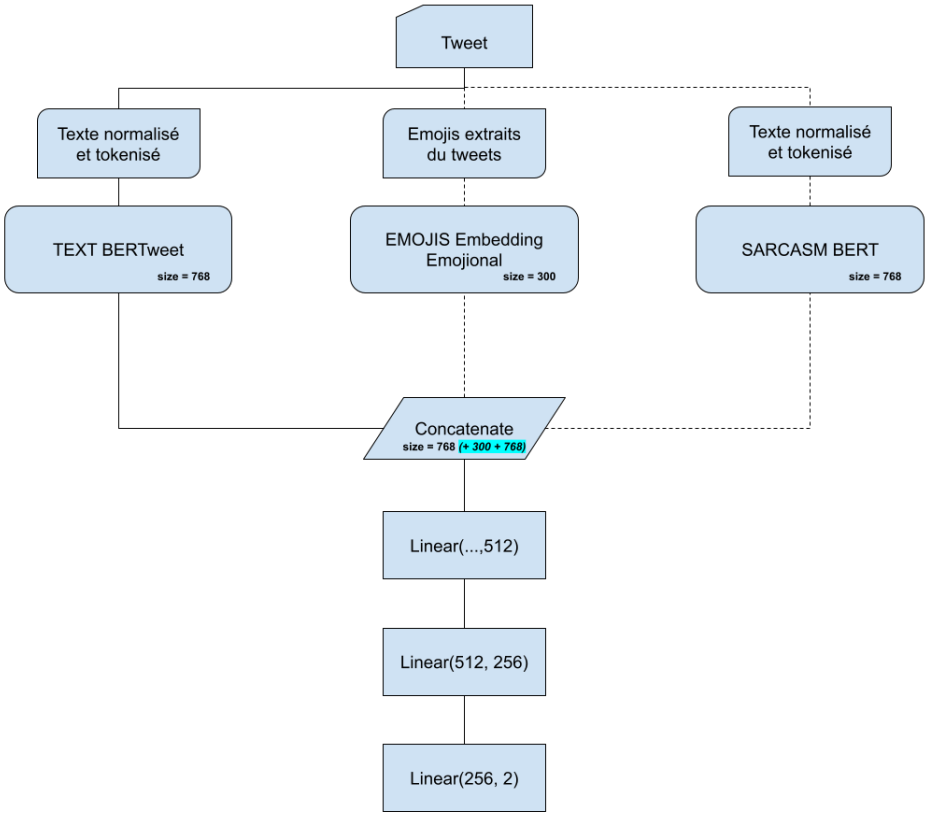

FIGURE 4.2 – Fusion de représentation utilisant la concaténation

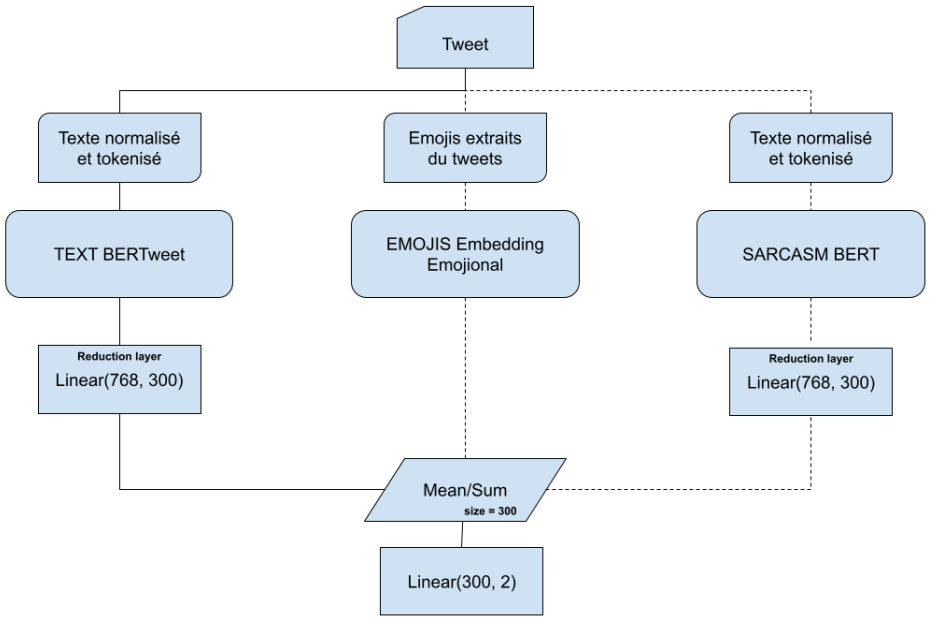

FIGURE 4.3 – Fusion de représentation utilisant la somme ou la moyenne

Les expérimentations suivantes seront donc réalisées avec cette couche d'*embedding* dégelée.

Concernant le mode de combinaison, le meilleur score F1 (86,28) ainsi que le meilleur écart-type sont atteints en utilisant la moyenne pour IberEval, même si les performances de la somme ne sont pas très loin derrière. Pour OffensEval, c'est la somme qui fournit les meilleures performances, avec un score F1 moyen de 84,26 et un écart-type de 1,14. Enfin, pour HatEval, la somme et la moyenne obtiennent les même scores F1 moyens, avec un écart-type très proche (la somme est inférieure à la moyenne de 0,03 point). Ce sont donc pour le moment la somme et la moyenne, dont le rang moyen est égal, qui nous permettent d'obtenir les meilleurs résultats. Ceci nous permet d'avoir un premier aperçu des méthodes de combinaison les plus efficaces pour ce type d'information.

	Baseline BERTweet			Concaténation		Somme		Moyenne	
	Score F1	std		Score F1	std	Score F1	std	Score F1	std
IberEval	79,62	2,18	Frozen	84,87	4,85	85,90	3,33	84,97	3,72
			Unfrozen	77,05	15,23	86,04	2,74	**86,28**	1,9
OffensEval	78,11	0,89	Frozen	83,76	3,82	85,56	4,3	81,89	4,72
			Unfrozen	81,78	3,89	**84,26**	1,14	84,00	4,75
HatEval	83,84	0,86	Frozen	88,32	3,13	**89,03**	0,98	86,9	4,76
			Unfrozen	87,44	6,44	86,73	5, 16	**89,03**	1,01
				Rang moyen : 3		Rang moyen : 1,33		Rang moyen : 1,33	

TABLE 4.2 – Scores F1 et écarts-types (std) pour la combinaison de BERTweet et Emojional. La *baseline* correspond à l'utilisation de BERTweet seul, après *fine-tuning*.

4.4.3 Inclure le sarcasme au classifieur de sentiments, avec ou sans prise en comte des emojis

Combiner BERTweet avec une représentation vectorielle d'emojis semble fonctionner correctement. Nous aimerions maintenant étudier l'impact de l'ajout du sarcasme sur l'analyse de sentiments, et ce incluant ou non la prise en compte des emojis. Si dans la littérature il y a des cas d'utilisation de l'information de sarcasme afin d'améliorer l'analyse de sentiments, les méthodes que nous employons sont différentes. En effet, nous avons tout d'abord tenté d'intégrer l'information de sarcasme de manière naïve, en tant que caractéristique, en nous inspirant de (Maynard and Greenwood, 2014) [96]. Les résultats obtenus lors de ces expérimentations préliminaires nous ont poussés à aller plus loin pour inclure le sarcasme à la classification de sentiments. Nous avons donc sélectionné un modèle à l'état de l'art permettant de faire de la détection de sarcasme, BERT, et avons utilisé son espace latent en 768 dimensions afin de le combiner à BERTweet, également en 768 dimensions, affiné sur l'analyse de sentiments. Ainsi, les différentes représentations seront fusionnées. Dans cette partie, nous verrons tout d'abord l'approche naïve utilisée pour intégrer le sarcasme en tant que caractéristique, puis nous nous concentrerons sur la combinaison des différentes représentations, en incluant ou non les emojis.

4.4.3.1 Approche naïve pour intégrer l'information de sarcasme dans la classification de sentiments

Dans la littérature, le sarcasme a parfois été intégré en tant que simple caractéristique afin d'effectuer de la détection de sentiments [96]. En nous inspirant de cette étude, nous avons voulu expérimenter une première méthode naïve pour prendre en compte l'information de sarcasme dans nos jeux de données.

En utilisant un classifieur de sarcasme, nous avons tout d'abord effectué une première étape d'annotation de nos jeux de données d'analyse de sentiments. Ceci nous a alors permis d'ajouter un tag « SARCASM » à la fin de chacun des tweets détectés comme étant sarcastique. La classification en sentiments des tweets est ensuite lancée, en prenant donc en compte l'information de sarcasme lorsque celle-ci a été détectée.

Afin de réaliser cette étape d'annotation, trois approches ont été comparées. Tout d'abord, un modèle T5 affiné sur de la détection de sarcasme sur Twitter et disponible sur HuggingFace [9] [120] a servi à la labellisation des jeux de données. Le T5 étant un modèle permettant la génération de texte, il s'agit donc de génération de label (*normal* ou *dérision*). Pour la seconde approche, nous avons affiné un BERT-base sur le jeu de données SemEval 2018 task 3a, spécialisé dans la détection d'ironie. Ce modèle a ensuite servi à annoter les corpus de sentiments avec le tag « SARCASM ». Enfin, pour la dernière approche, ce sont cette fois 10 modèles BERT-base qui ont été affinés sur les données de SemEval 2018 task 3a. Un vote à la majorité est ensuite appliqué afin de déterminer si chacun des tweets est sarcastique ou non.

Le tableau 4.3 présente les résultats de l'analyse de sentiments avec et sans information de sarcasme, sur trois jeux de donneés. Afin de comparer les deux approches, les scores f1 ainsi que les écarts-types sont analysés. Comme nous pouvons le voir, pour les jeux de données IberEval et HatEval (les deux corpus contenant deux classes), la présence d'information de sarcasme ne permet pas d'améliorer la classification de sentiments. En effet, les scores f1 restent similaires : seulement un gain de 0,29 point en utilisant le vote à la majorité pour l'annotation en sarcasme. Avec le T5, l'écart-type est légèrement réduit (-0,55 point), le modèle généralise mieux. Concernant le jeu de donnée SemEval 2016 task 6, nous pouvons constater une dégradation des performances de classification de sentiments : quel que soit la méthode utilisée pour l'annotation en sarcasme, le score f1 est diminué ; -1,02 points avec le vote à la majorité, et jusqu'à -2,32 points en utilisant un modèle aléatoire affiné sur SemEval 2018. En revanche, l'écart-type reste inchangée.

Les résultats obtenus ne montrent aucun progrès de la détection de sentiments en ajoutant l'information de sarcasme avec cette approches naïve. Nous avons donc souhaité tenter une autre méthode d'intégration du sarcasme, en reprenant les méthodes de combinaisons expérimentée pour l'intégration des emojis. Dans les parties suivantes, nous voyons donc les résultats de ces combinaisons.

9. https://huggingface.co/mrm8488/t5-base-finetuned-sarcasm-twitter

	HatEval		IberEval		SemEval 2016	
	Avec sarc.	Sans sarc.	Avec sarc.	Sans sarc.	Avec sarc.	Sans sarc.
BERT affiné	81,73 (±1,19)		78,47 (±2,78)		57,68 (±3,1)	
10 BERT affinés	81,75 (±1,06)	81,76 (±0,94)	78,88 (±2,97)	78,59) (±2,85)	58,98 (±2,85)	60,00 (±2,9)
T5 sarc.	81,76 (±1,06)		78,56 (±2,3)		58,70 (±2,72)	

TABLE 4.3 – Scores F1 et écarts-types (std) pour l'intégration naïve du sarcasme grâce à un tag

4.4.3.2 Inclure le sarcasme au sentiment, sans les emojis

Puisque l'approche naïve n'a pas fonctionné, nous avons cette fois utilisé le modèle BERT afin d'avoir une représentation de sarcasme, et nous l'avons combiné à BERTweet. Plus précisément, nous avons extrait l'espace latent d'un modèle BERT affiné sur un corpus de sarcasme, et fusionné les 768 dimensions avec celles de BERTweet affiné sur de la classification de sentiments.

Les résultats des expérimentations concernant BERTweet combiné aux informations de sarcasme sont présentés dans le tableau 4.4. Bien que ces résultats soient inférieurs à ceux de la combinaison BERTweet+Emojis (voir tableau 4.2), ils restent supérieurs à la *baseline*. En effet, nous pouvons observer un gain de 3,18 et 3,39 points de score F1 moyen en utilisant respectivement la concaténation et la moyenne sur IberEval, ainsi qu'un gain de 4,91 points de score F1 moyen sur OffensEval en utilisant la somme pour combiner les deux éléments. Cette fois encore, nous pouvons constater que les meilleures performances sont atteintes pour les deux jeux de données en utilisant la somme et la moyenne. Pour HatEval, c'est également la somme qui permet d'obtenir les meilleurs scores F1, avec un gain de 1,47 points par rapport à la *baseline*.

Les résultats du tableau 4.4 confirment la conclusion de l'expérimentation précédente : c'est une nouvelle fois la somme et la moyenne, deux méthodes mathématiquement proches, qui obtiennent les meilleurs résultats. Nous pouvons donc émettre l'hypothèse que ce type de combinaison permet au mieux de conserver les informations contenues dans les différentes caractéristiques combinées (BERTweet, BERT sarcasme et Emojional) comparées à la concaténation.

Nous avons pu voir précédemment que l'utilisation des emojis améliorait les performances du modèle. Nous venons maintenant de voir que la combinaison d'une représentation de sarcasme permettait également d'améliorer la classification. Nous pouvons donc émettre l'hypothèse que fusionner à la fois la représentation d'emojis et BERT sarcasme avec BERTweet devrait permettre d'obtenir d'encore meilleurs scores, en donnant ainsi au modèle toutes les informations nécessaires pour qu'il puisse correctement prédire le sentiment.

4.4.3.3 Inclure les emojis et le sarcasme au sentiment

Notre objectif est d'étudier l'impact que peut avoir l'ajout à la fois d'information d'emojis et de sarcasme sur notre classifieur de sentiments. En effet, notre modèle de sarcasme et la représentation d'emojis ont individuellement amélioré notre classifieur de sentiments. Nous avons donc l'intuition qu'ajouter ces deux types d'information en

	Baseline		Concaténation		Somme		Moyenne	
	Score F1	std	Score F1	std	Score F1	std	Score F1	std
IberEval	79,62	2,18	82,80	3,89	80,93	2,52	**83,01**	3,07
OffensEval	78,11	0,89	79,64	2,89	**83,02**	3,66	80,71	3,31
HatEval	83,84	0,86	84,04	2,19	**85,31**	2,69	84,74	2,7
			Rang moyen : 2,67		Rang moyen : 1,67		Rang moyen : 1,67	

TABLE 4.4 – Scores F1 et écarts-types (std) pour la combinaison de BERTweet et du BERT sarcasme

même temps devrait permettre d'améliorer l'analyse de sentiments.

L'idée ici est donc de combiner trois choses : un modèle affiné sur de l'analyse de sentiments (BERTweet [104]), une représentation vectorielle d'emojis (Emojional [8]), et un modèle affiné sur de la détection de sarcasme (BERT [33]). Ces trois représentations seront encore une fois combinées en utilisant les trois méthodes utilisées précédemment. Les trois vecteurs seront donc soit concaténés en un grand vecteur de 1836 dimensions (768 pour BERTweet + 768 pour BERT sarcasme + 300 pour Emojional), soit moyennés ou sommés. Pour la moyenne et la somme, BERTweet et BERT doivent au préalable subir une étape de réduction du nombre de leur dimensions afin d'atteindre les 300 dimensions de Emojional. Chaque réduction est effectuée au moyen d'une couche linéaire (*Dense*), qui prend en entrée le vecteur de 768 dimensions, et donne en sortie un vecteur de 300 dimensions. Une fois que les trois représentations possèdent le même nombre de dimensions, il est possible d'effectuer une moyenne ou une somme, résultant en un vecteur final de 300 dimensions.

Les résultats de ces expérimentations, ainsi que des précédentes, sont réunies dans le tableau 4.5. Comme nous pouvons le voir, la combinaison des trois éléments (ALL) obtient de meilleurs scores que la *baseline* (jusqu'à +2,91 point), malgré une perte en généralisation dans tout le tableau de manière générale. Si pour IberEval, c'est la concaténation qui réalise les meilleures performances, c'est une fois de plus la somme et la moyenne qui fonctionnent le mieux pour les autres jeux de données, respectivement +2,91 points pour HatEval et +2,57 pour OffensEval.

En comparant ces scores avec ceux obtenus précédemment (BERTweet+Emojis et BERTweet+Sarcasme), nous pouvons voir que les scores F1 sont bien inférieurs lors de la combinaison des trois éléments. Ceci est particulièrement vrai pour le jeu de données OffensEval, avec lequel les performances de ALL sont inférieures quelle que soit la méthode de combinaison. Au vu des résultats obtenus avec BERTweet+Emojis et BERTweet+sarcasme, notre première hypothèse est que le détecteur de sarcasme empêche le modèle d'atteindre son potentiel maximum, puisque BERTweet+sarcasme a des performances moins élevées que BERTweet+emoji. Ceci pourrait être dû à des informations contradictoires apportées par le classifieur de sarcasme. Comme nous avons pu le voir précédemment, la combinaison du modèle de sentiments avec la représentation d'emojis obtient toujours les meilleurs scores F1, mis à part la concaténation pour le jeu de données IberEval. Ceci peut s'expliquer par le fait qu'il s'agisse de notre plus petit jeu de données, le modèle peut donc avoir du mal à apprendre, en

particulier pour des couches linéaires permettant la classification qui n'ont pas eu de pré-apprentissage.

Notre seconde hypothèse concerne plutôt la manière dont les éléments sont combinés. En effet, il est possible que des méthodes de combinaison aussi simple ne soient pas efficaces pour prendre en compte les différentes informations. Afin de vérifier cela, nous proposons de poursuivre nos expériences en utilisant une méthode de combinaison différente, ainsi qu'en réalisant une *ablation study*, c'est-à-dire en retirant le classifieur de sentiments, et en conservant uniquement Emojional et BERT sarcasme. Ceci nous permettra de tester l'impact de la représentation d'emojis et du modèle de sarcasme seuls, sans BERTweet.

	Baseline		Elt Comb.	f-uf	Concaténation		Somme		Moyenne	
	Score F1	std			Score F1	std	Score F1	std	Score F1	std
IberEval	79,62	2,18	BERTweet+Emojis	Unfrozen	77,05	15,23	86,04	2,74	**86,28**	1,9
			BERTweet+Sarc.	-	82,80	3,89	80,93	2,52	83,01	3,07
			ALL	Unfrozen	82,53	3,97	81,49	2,78	80,80	3,16
OffensEval	78,11	0,89	BERTweet+Emojis	Unfrozen	81,78	3,89	**84,26**	1,14	84,00	4,75
			BERTweet+Sarc.	-	79,64	2,89	83,02	3,66	80,71	3,31
			ALL	Unfrozen	78,3	2,7	80,68	3,62	79,10	2,72
HatEval	83,84	0,86	BERTweet+Emojis	Unfrozen	87,44	6,44	86,73	5,16	**89,03**	1,01
			BERTweet+Sarc.	-	84,04	2,19	85,31	2,69	84,74	2,7
			ALL	Unfrozen	84,39	2,88	86,73	3,14	85,41	2,59

TABLE 4.5 – Scores F1 et écarts-types (std) pour la combinaison de BERTweet, BERT sarcasme et Emojional

4.4.4 Étude approfondie

En souhaitant étudier l'impact des caractéristiques d'emojis et de sarcasme sur la classification de sentiments, nous avons mis en avant une contradiction grâce à nos précédentes expérimentation. En effet, l'ajout d'une représentation des emojis à la classification de sentiments améliore le modèle, de même que la combinaison avec un modèle de détection de sarcasme. Cependant, l'utilisation de ces trois informations simultanément semble freiner les performances du modèle final. Deux hypothèses ont alors été proposées : les emojis entrent en conflit avec le modèle de sarcasme, ou bien le mode de combinaison des éléments est trop naïf. Dans cette partie, nous étudierons donc ce comportement à travers une étude du mode de combinaison, suivi d'une *ablation study*.

4.4.4.1 Étude sur le mode de combinaison

Afin d'analyser l'apport du mode de combinaison sur la qualité de l'apprentissage, nous proposons de comparer les approches naïves de combinaison avec les approches hybrides suivantes :

— une première approche, pour laquelle le nombre de dimensions de BERTweet est réduit afin de moyenner le modèle avec la représentation des emojis, puis cette moyenne est concaténée avec BERT sarcasme, réduit également en dimension. Puis la classification finale est effectuée (figure 4.4).

— une seconde, dont l'architecture est proche de celle décrite précédemment, cependant le nombre de dimensions est tout d'abord augmenté, avant d'appliquer une fonction d'activation (ReLU) avant chaque combinaison (figure 4.5).

En appliquant différentes méthodes combinaisons ensemble, nous souhaitons tester deux choses : tout d'abord, si un mode de combinaison plus complexe permet d'améliorer l'analyse de sentiments, mais également si l'ajout de couches non linéaire (fonction d'activation) a un impact sur l'agrégation des informations.

Nous pouvons voir tableau 4.6 que la version classique, c'est-à-dire sans augmentation ni fonction d'activation (figure 4.4) a des performances similaires à la moyenne classique des trois informations différentes (ALL) en tableau 4.5. Ces résultats restent donc inférieurs à l'utilisation de la représentation d'emojis seule avec le classifieur de sentiments. En revanche, l'ajout d'augmentation de dimension et de ReLU (figure 4.5) améliore le score F1 de ALL. Notre hypothèse est donc que les espaces de projection sont trop éloignés/différents pour être combinés de manière naïve. Ce problème est légèrement corrigé par l'augmentation puis réduction du nombre de dimensions, et surtout par l'utilisation de couches non linéaires, ici des ReLU.

	Elt Comb.	Concaténation		Somme		Moyenne		Moyenne+Concat.		Moyenne+Concat.+ReLU	
		Score F1	std	Score F1	std	Score F1	std	Score F1	std	Score F1	std
IberEval	BERTweet+Emojis	77,05	15,23	86,04	2,74	86,28	1,9	80,68	2,96	83,2	3,44
	BERTweet+Sarc.	82,80	3,89	80,93	2,52	83,01	3,07				
	ALL	82,53	3,97	81,49	2,78	80,80	3,16				
OffensEval	BERTweet+Emojis	81,78	3,89	84,26	1,14	84,00	4,75	78,7	2,51	82,93	2,67
	BERTweet+Sarc.	79,64	2,89	83,02	3,66	80,71	3,31				
	ALL	78,3	2,7	80,68	3,62	79,10	2,72				
HatEval	BERTweet+Emojis	87,44	6,44	86,73	5,16	89,03	1,01	84,88	2,92	83,49	2,22
	BERTweet+Sarc.	84,04	2,19	85,31	2,69	84,74	2,7				
	ALL	84,39	2,88	86,73	3,14	85,41	2,59				

TABLE 4.6 – Scores F1 et écarts-types (std) pour la combinaison de BERTweet, BERT sarcasme et Emojional, en utilisant la concaténation et la moyenne (voir schémas 4.4 et 4.5)

4.4.4.2 Ablation study

Les résultats obtenus montrent que l'ajout des caractéristiques de sarcasme et d'emojis permettent d'obtenir de meilleurs scores qu'en utilisant le texte seul (*baseline*). Cependant, cette amélioration reste inférieure à l'utilisation du texte avec les informations d'emojis ou de sarcasme individuellement. Afin d'étudier plus en détail l'impact de chaque élément et de déterminer l'influence de chacune des caractéristiques, nous avons réalisé des expérimentations supplémentaires. En effet, dans l'idée d'examiner plus précisément les caractéristiques d'emojis et de sarcasme, nous voulons vérifier ce qu'il se passe en utilisant uniquement une seule des caractéristiques ajoutées. Cela signifie que la l'*embedding* d'emojis Emojional et BERT sarcasme seront utilisés sans le modèle de sentiments BERTweet, seuls ou combinés ensemble de manière naïve (c'est-à-dire une utilisation à la fois de Emojional et de BERT sarcasme).

Emojis + Dense

Dans les résultats précédents, nous avons pu voir qu'utiliser un *embedding* d'emojis ou un modèle de sarcasme en complément d'un modèle d'analyse de sentiments permettait

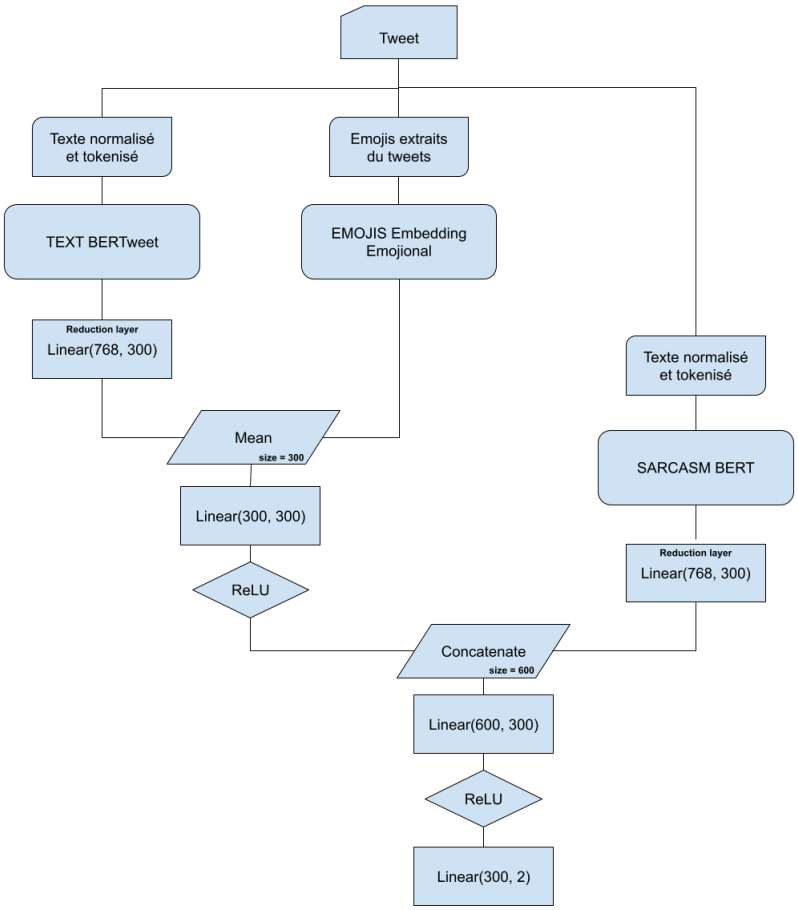

FIGURE 4.4 – Architecture utilisée pour la combinaison de BERTweet, BERT sarcasme et Emojional simple

d'améliorer la classification. Nous avons également pu constater qu'ajouter conjointement ces deux types d'informations améliorait plus faiblement le modèle d'analyse de sentiments. Dans cette partie, nous étudions l'impact de l'*embedding* d'emojis sur la classification du sentiment pour nos jeux de données. Les résultats de ces expérimentations sont contenus dans le tableau 4.7. Nous pouvons voir que le score F1 est particulièrement mauvais sur le jeu de données complet. Cela s'explique par le fait que tous les tweets ne contiennent pas d'emojis. Ainsi, certains des tweets ont pour représentation une suite d'emojis vides. Compte tenu de la faible proportion de tweets avec emojis, nous n'avons pas reproduit cette expérimentations uniquement sur les tweets qui en contiennent. En effet, IberEval ne contient que 231 tweets avec des emojis (296 pour HatEval et 699 pour OffensEval), ce qui est trop peu pour affiner un modèle.

Sarcasme + Dense

Après avoir vu que l'utilisation d'une représentation des emojis seule ne permettait pas la classification du sentiment, nous allons maintenant tester notre modèle de détection de sarcasme seul sur notre tâche. Pour rappel, le modèle en question est un BERT-base affiné sur le corpus SemEval 2018 task 3, c'est-à-dire de la détection de sarcasme. Le

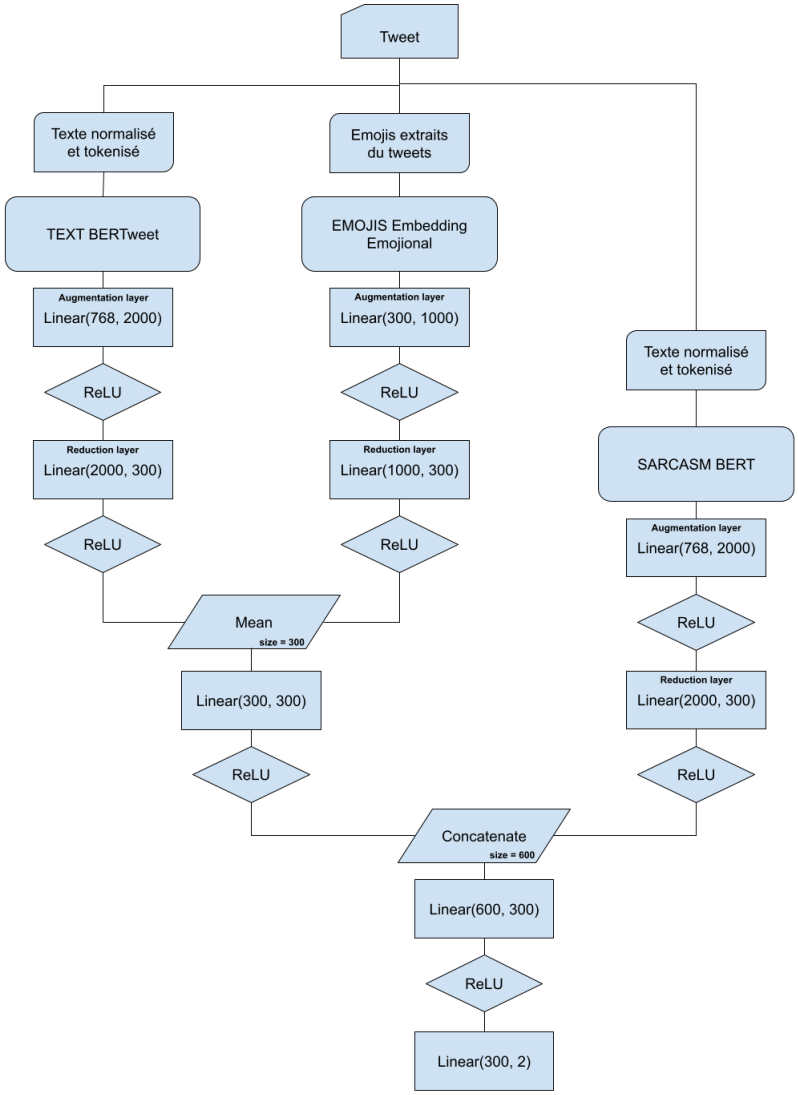

FIGURE 4.5 – Architecture utilisée pour la combinaison de BERTweet, BERT sarcasme et Emojional, avec augmentation de dimensions et fonction d'activation

tableau 4.7 réunit ces résultats. Nous pouvons voir que le score est déjà plus semblable aux scores F1 précédents. Notre hypothèse est que le *fine-tuning* de sarcasme est probablement oublié, et BERT se transforme alors en simple classifieur de sentiments, ce qui peut expliquer des scores proches de la *baseline*. Le modèle BERT n'étant pas spécialisé sur les données Twitter, nous obtenons donc des résultats proches, voir légèrement en dessous de la *baseline*.

Emojis + Sarcasme + Dense

Enfin, nous avons voulu tester l'analyse de sentiments en utilisant à la fois le modèle de sarcasme et la représentation des emojis ensemble. Afin de combiner ces deux types de caractéristiques, nous nous sommes servis de la somme et de la moyenne, car ces deux méthodes ont obtenu les meilleures performances sur nos précédentes expériences. Les résultats de ces expérimentations sont notés dans le tableau 4.7. Avec Emojional

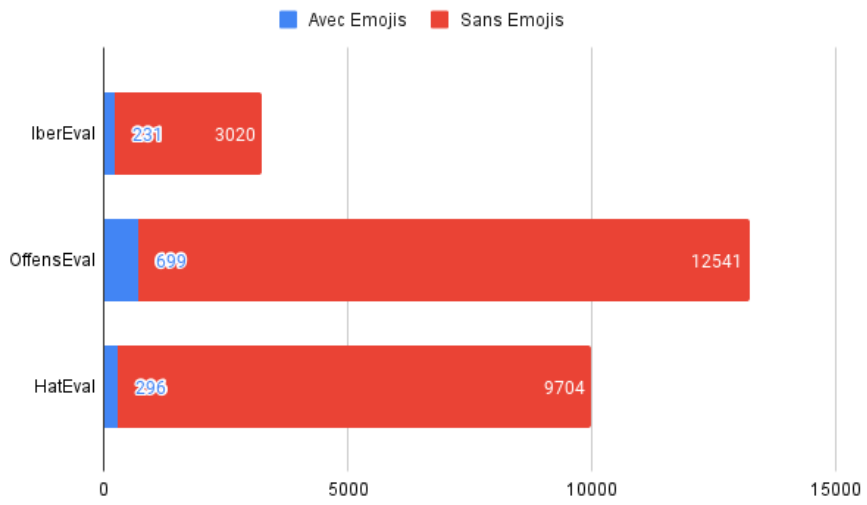

FIGURE 4.6 – Répartition des tweets contenant des emojis pour chaque jeu de données

combiné au modèle de sarcasme, nous pouvons voir qu'il n'y a pas d'amélioration liée à l'ajout des emojis. En effet, les scores obtenus sont proches de ceux du sarcasme seul. Une fois encore, le modèle de sarcasme se transforme en modèle de classification de sentiments, oubliant son précédent *fine-tuning*, et les performances sont similaires à la *baseline*.

		Score F1	std
IberEval	Emojis + Dense	36,93	0,73
	Sarcasme + Emojis + Dense (mean)	79,02	1,14
	Sarcasme + Dense	79,19	1,13
OffensEval	Emojis + Dense	40,59	0,35
	Sarcasme + Emojis + Dense	77,41	0,41
	Sarcasme + Dense	77,37	0,34
HatEval	Emojis + Dense	37,06	0,31
	Sarcasme + Emojis + Dense	82,25	0,68
	Sarcasme + Dense	82,49	0,69

TABLE 4.7 – Résultats de l'*Ablation study*, Scores F1 et écarts-types (std) pour la classification utilisant Emojional ou BERT sarcasme seuls, ou une combinaison des deux (sans BERTweet).

Analyse d'erreur de ALL

Afin de mieux comprendre l'impact du classifieur de sarcasme sur le modèle d'analyse de sentiments, nous avons tout d'abord réalisé une exploration manuelle des cas d'erreurs de ALL. Nous avons ainsi pu constater que ces tweets sont particulièrement ambigus et difficiles à classifier, même pour un humain. Ceci est lié à la nature des jeux de données, qui traitent la haine, la misogynie et l'agressivité. Dans un second temps, nous avons voulu comparer les prédictions pour certaines de nos expérimentations. Nous avons donc vérifié les erreurs de ALL lors de la prédiction du sentiment, et

	% de tweets sarcastiques parmi les erreurs de ALL
IberEval	59,05 %
OffensEval	60,36 %
HatEval	63,96 %

TABLE 4.8 – Étude des cas d'erreur de ALL

observé si les tweets étaient détectés comme étant sarcastiques ou non. Les résultats de cette étude sont disponibles dans le tableau 4.8. Il s'avère que dans la majorité des cas, les tweets mal classifiés par ALL sont sarcastiques (63,96% des cas pour HatEval). Il y a donc bien un conflit entre les informations apportées par le classifieur de sarcasme, et celles du modèle de sentiments et des emojis. La partie suivante nous permettra d'effectuer une analyse globale de nos résultats, et de proposer des pistes pour solutionner le problème de conflit.

4.4.5 Analyse des résultats

Nous avons réalisé une étude permettant d'étudier l'impact de l'ajout de caractéristiques d'emojis et de sarcasme à notre classifieur de sentiments. Afin de trouver la bonne méthode pour combiner nos caractéristiques avec notre modèle, nous avons dans le même temps réalisé une étude comparative de trois différents modes de combinaison : la concaténation, la somme et la moyenne.

Comme nous pouvons le voir dans le tableau 4.5, les meilleures performances pour chacun des jeux de données sont obtenues avec BERTweet+Emojis. Les scores F1 pour l'utilisation du sarcasme en complément du modèle de SA, bien que tout de même au-dessus de la *baseline*, sont inférieurs à ceux avec les emojis. Afin de maximiser l'adaptation de l'*embedding* d'emojis aux données, il est plus judicieux de dégeler la couche d'*embedding*. En effet, pour deux de nos trois corpus, en ayant la couche d'*embedding* dégelée, les performances atteignent +6,66 de score F1, et -0,28 d'écart-type, par rapport à la *baseline* BERTweet seul pour IberEval et +6,15 de score F1 et +0,25 d'écart-type pour OffensEval. Ces performances sont obtenues en utilisant la moyenne ou la somme comme moyen de combinaison. En effet, ces deux modes de combinaison semblent agréger au mieux nos différents types d'information en limitant les pertes. Notre hypothèse est donc que la concaténation de grands vecteurs (768 pour BERTweet, 768 pour BERT et 300 pour Emojional) limite l'apprentissage du modèle. Cette hypothèse sera vérifiée ultérieurement, les résultats ne seront donc pas présentés dans cette thèse.

Concernant les scores moins élevés de la combinaison des trois modèles, deux hypothèses ont été émises. La première concerne la manière dont les différentes informations sont combinées. En effet, en tentant de mélanger à la fois de l'analyse de sentiments, une représentation des emojis et un détecteur de sarcasme, il est possible qu'un mode de combinaison aussi naïf que la somme ou la moyenne ne soit pas suffisant pour que toutes les informations soient assimilées par le modèle final. Pour tester cela, nous avons comparé deux nouveaux modes de combinaison de ces trois éléments. Si cela nous a permis d'améliorer les performances du modèle, les scores restent tout de même en

dessous de l'utilisation de BERTweet+Emojis. Notre seconde hypothèse concerne donc un potentiel conflit entre les informations apportées par les emojis et par le détecteur de sarcasme. Pour la tester, nous avons tout d'abord effectué une *ablation study*. Plus précisément, nous avons voulu voir plus en détail comment réagissaient le modèle de sarcasme et la représentation des emojis lorsqu'ils étaient utilisés seuls, ou simplement combinés entre eux. Pour Emojional, les jeux de données ne contiennent pas assez de tweets pour que la représentation vectorielle seule permette la classification. En revanche, pour le modèle BERT affiné sur la détection de sarcasme, seul ou avec la représentation des emojis, le modèle atteint des performances proches de la *baseline*. Il s'avère que notre modèle de sarcasme se transforme alors en modèle d'analyse de sentiments, oubliant rapidement son précédent *fine-tuning*. La dernière expérience menée nous a permis d'obtenir une réponse partielle à nos hypothèses. En effet, nous avons voulu directement analyser si le sarcasme avait un impact négatif sur la combinaison des trois éléments. Nous avons, dans premier temps, observé manuellement les erreurs du modèle, et remarqué que la plupart de ces tweets étaient difficiles à classifier, même pour un humain. Une solution à cela pourrait être l'ajout d'une classe supplémentaire « indécidable », qui serait utile pour les jeux de données traitant des thématiques compliquées, comme la misogynie, la haine et l'agressivité. Dans un second temps, nous avons vérifié la classification de sarcasme pour ces tweets mal classifiés. Au vu des résultats de notre étude approfondie, les informations apportées par la représentation vectorielle des emojis combinée au modèle de sentiments BERTweet semblent entrer en contradiction avec le modèle de détection de sarcasme puisqu'en moyenne 61,12 % des erreurs du modèle sont des tweets sarcastiques.

En effet, le modèle final combinant les trois éléments a plus de difficulté à classifier les tweets identifiés comme étant sarcastiques. D'autres expérimentations sont donc à mener, notamment concernant l'ajout de pondération apprise par le modèle, lui permettant de donner plus ou moins d'importance à chaque modèle ou représentation selon le tweet. Par exemple, un tweet ne contenant pas d'emojis ne nécessitera pas d'information provenant de la représentation vectorielle des emojis. Le temps étant limité, ces expérimentations seront poursuivies dans un second temps et non présentées dans cette thèse, afin d'apporter des clarifications à nos hypothèses. Ainsi, nous recommandons donc pour le moment l'utilisation d'une représentation vectorielle d'emojis afin d'apporter des informations supplémentaires sur les tweets. Afin de les combiner, la somme et la moyenne semblent être plus performantes (il s'agit de toute manière de deux méthodes très proches).

Chapitre 5

Projet SAPhIRS et l'industrialisation de l'apprentissage automatique

Résumé

Ce chapitre est dédié au projet SAPhIRS réalisé en partenariat entre Saagie, le LITIS et Airbus Defence and Space, et dans lequel s'inscrit cette thèse. L'objectif de ce projet est la détection de radicalisation, d'influenceurs et de communautés. Nous commençons par présenter le projet, ainsi que les différents partenaires, puis nous détaillons nos différentes contributions pour SAPhIRS. Dans ce chapitre, nous décrivons également notre participation à la compétition SemEval 2019, effectué dans le cadre du projet, dans l'attente de la collecte d'un jeu de données de radicalisation.

Sommaire

Projet SAPhIRS et l'industrialisation de l'apprentissage automatique

L'avènement des réseaux sociaux a conduit à une large augmentation de données publiées sur Internet. Cela concerne notamment l'émission d'opinions sur des événements de la vie quotidienne, publique et politique, ou la diffusion d'idéologie sur des sujets parfois sensibles (appel à la haine, discours radicaux, etc.). Cette thèse s'inscrit dans le projet SAPhIRS, afin d'analyser ces phénomènes. Ainsi, dans le but d'étudier les mécanismes de propagation de l'information et de l'opinion au sein des réseaux sociaux, le projet SAPhIRS, pour « Système pour l'Analyse de la Propagation d'Information dans les Réseaux Sociaux », a été mis en place. Il s'agit d'un projet inscrit dans le dispositif RAPID, retenu par le comité de sélection conjoint tenu par la DGA et DGE en fin 2017.

Ce projet a été développé par Saagie, porteur du projet, en partenariat avec le LITIS et Airbus Defence and Space. Les travaux décrits dans ce chapitre ont été réalisés par les trois partenaires, il s'agit donc d'un travail collectif. Mes contributions pour ce projet (en collaboration avec l'équipe Recherche de Saagie) concernent principalement la création de modules pour le pipeline de détection de haine (collecte et stockage des tweets, détection de langue et de dialecte, translittération, détection de haine, d'objet d'opinion et de polarité d'opinion), ainsi que la mise en place du pipeline et la visualisation des différents résultats obtenus (voir tableau 5.6). Cette thèse étant réalisée dans le cadre du dispositif CIFRE avec le laboratoire LITIS ainsi que dans l'entreprise Saagie, les différentes contributions auxquelles j'ai pu participer correspondent à la fois à des contributions de Saagie et du LITIS.

Le projet SAPhIRS a pour objectif d'élaborer des outils de détection de phénomènes de radicalisation dans les réseaux sociaux, et plus précisément sur Twitter. Pour répondre à cette problématique, nous proposons de développer un outil permettant l'analyse d'opinion, la reconnaissance de changements d'opinion, ainsi que la détection de *leaders* d'influence dans le domaine civil et de la sécurité. Les travaux sont focalisés sur deux cas d'utilisation : le domaine civil et militaire. Dans le domaine civil, il s'agit d'analyser des opinions liées au tourisme en Normandie, ainsi que leur propagation. Dans le domaine militaire, il est question cette fois de détecter et d'analyser des messages radicaux ou d'appel à la haine, de suivre leur propagation et de détecter les *leaders* d'influence.

Le terme radicalisation est ici employé au sens large, c'est-à-dire qu'il s'agit du « processus d'adoption d'une croyance extrémiste incluant la volonté d'utiliser, de soutenir ou de faciliter la violence comme méthode de changement de la société » [1]. Cette définition comprend notamment la radicalisation religieuse (ainsi que le terrorisme), ou le hooliganisme pour le domaine du sport.

Afin de faciliter son développement, le projet a été divisé en trois lots. Le premier lot implique directement Saagie, puisqu'il concerne la collecte et le stockage de données,

1. http://radicalisation.fr/radicalisation_definition.php

l'entraînement de modèles et la démonstration grâce à la solution Saagie. Le lot 2 consiste à implémenter différents modules permettant l'analyse d'opinion et la détection de radicalisation sur Twitter. Enfin, le lot 3 est lié aux solutions permettant de détecter la propagation d'influences.

Si collecter des données Twitter est simple, grâce à l'API officielle [2] du réseau social, l'accès et surtout l'annotation de tweets radicaux est plus difficile. Pour faciliter la création de ce jeu de données, nous sommes passés par l'entreprise externe ELDA [3]. Ceci nous a permis d'obtenir un corpus final de 100 000 tweets annotés en objet d'opinion et en polarité.

Cependant, la constitution de ce jeu a pris du temps. Dans l'attente de l'obtention du corpus spécialisé sur la radicalisation, avec l'équipe Recherche de Saagie, nous avons pris part à la tâche 5 de la compétition SemEval 2019 [9] : la détection de haine. Ceci nous permit de tester différentes approches sur une thématique proche de la nôtre.

Ce chapitre est consacré aux différentes étapes du projet SAPhIRS et à sa mise en production. Nous y voyons donc les différentes étapes de sa construction. Nous commençons par présenter plus en détail le projet, avant d'aborder la partie dédiée à la détection d'influences et de communautés (lot 3). Nous traitons ensuite les modules créés par Saagie, qui correspondent principalement aux lots 1 et 2, avant de voir la création du *pipeline* final, le déploiement et la visualisation des résultats. Enfin, nous terminons par décrire la participation à la compétition SemEval 2019, avant de présenter un tableau récapitulatif de mes contributions au projet.

5.1 Présentation du projet SAPhIRS

Comme nous l'avons dit précédemment, le but du projet SAPhIRS est de développer des outils permettant de détecter la radicalisation sur Twitter. Le projet est réalisé en partenariat entre Saagie, le LITIS et Airbus Defence and Space et est découpé en plusieurs lots afin de faciliter son avancement.

5.1.1 Partenariat entre Saagie, le LITIS et Airbus Defence and Space

Saagie, est une entreprise créée en 2013. Elle accompagne les entreprises dans leur transition numérique en les libérant des barrières technologiques liées au Big Data. En plus de ses activités de service, qui l'ont amené à travailler avec de nombreux acteurs français et internationaux (Vente privée, Johnson&Johnson, Icade, Caisse d'Epargne Normandie, Ferrero, Bouygues Energies Services, Matmut...) Saagie a lancé sa propre plateforme Big Data. Clé en main, de bout en bout et orientée métier, elle permet d'exploiter pleinement la richesse des données des entreprises de tous secteurs d'activité souhaitant placer la donnée au cœur de leur culture, leur permettant ainsi de créer de

2. https://developer.twitter.com/en/docs
3. http://www.elda.fr/en/

nouveaux leviers de croissance. Ambassadeur de la Normandy French Tech, hébergée au sein de Seine Innopolis, l'écosystème normand du numérique, Saagie est soutenue par la Région Normandie et la BPI et accompagnée par The Family, le Réseau Entreprendre et le Hub BPI France. Saagie se compose d'une équipe pluridisciplinaire : *data scientists*, développeurs, architectes Data ou encore experts métier, ainsi qu'un Pôle Recherche. Le pôle Recherche de Saagie, se concentre sur trois thématiques : l'apprentissage faiblement supervisé, l'apprentissage multimodale, et l'explicabilité des modèles. Il a également pour objectif de participer à l'intégration au produit Saagie de nouvelles technologies liées à l'apprentissage profond, ainsi qu'au rayonnement de Saagie via des compétitions.

Le laboratoire d'informatique, de traitement de l'information et des systèmes (LITIS, EA 4108) est l'unité de recherche en sciences et technologies de l'information de Haute-Normandie commune à l'INSA de Rouen, à l'Université de Rouen et à celle du Havre. Le LITIS développe des démarches cohérentes pour mieux comprendre et maîtriser la nature de l'information et de son utilisation contextuelle. Ses recherches portent à la fois sur des aspects théoriques, algorithmiques et sur la mise en œuvre de systèmes sensibles au contexte, allant du capteur à la base de données. Deux équipes sont impliquées dans le laboratoire : l'équipe MIND (Multi-agents, Interaction, Décision), dont les travaux portent sur la conception de systèmes de décision automatiques ou semi-automatiques et sur les interactions médiées entre utilisateurs humains, et l'équipe App (Apprentissage) qui étudie les techniques de modélisation et d'apprentissage statistique permettant d'appréhender la diversité des données (dimensionnalité, structures, non-stationnarité) et la nature des solutions attendues (connaissances *a priori*).

Depuis janvier 2021, Saagie et le LITIS ont monté ensemble le Labcom L-Lisa, impliquant à la fois les ingénieurs de Saagie et les chercheurs du LITIS. Avec ce laboratoire commun, l'objectif de développer des méthodologies déclinées autour de deux axes. Dans un premier temps, il s'agit de traiter des données médicales en collaboration avec le centre de lutte contre le cancer Henri Becquerel. Dans un second temps, cela concerne le traitement de données utilisateurs du produit Saagie.

Le département Traitement Avancé de l'Information d'Airbus Defence and Space réalise des études amont à caractère technique et/ou opérationnel et des démonstrateurs appliqués pour des clients tels que le Ministère de la Défense, le Ministère de l'Intérieur, la Commission Européenne ou l'Agence de Défense Européenne. Ce département a eu la charge de programme d'étude amont (PEA) tels que HERISSON sur l'évaluation des technologies de traitement de l'information non structurée, MAURDOR, sur la reconnaissance de documents écrits, TRAD sur la traduction automatique de texte et de la parole. Il participe à de nombreux projets partiellement financés par la commission Européenne ou l'Agence nationale de la Recherche. Ce département développe également en grande partie sur fonds propre la plateforme WebLab qui a été utilisée dans ces différents projets. Enfin, le département accueille régulièrement des doctorants CIFRE.

5.1.2 Découpage du projet

Dans le cadre du projet SAPhIRS, le traitement de la donnée se découpe en trois axes complémentaires. Le premier axe (lot 1) concerne la mise en place d'un socle technique permettant la collecte et l'annotation de données, l'entraînement de modèles, et la démonstration par des interfaces graphiques. La mise en place de ce socle fondé sur la solution Saagie permettra d'itérer rapidement les lot 2 et 3 sur l'ensemble de la durée du projet. Le deuxième axe correspond à un module d'analyse d'opinions et de détection de radicalisation sur les réseaux sociaux (lot 2). Quant au troisième, il s'agit d'un modèle de propagation d'opinions sur Twitter, permettant la détection de *leaders* d'opinion et de tentatives d'influence (lot 3). Les données collectées sur la plateforme Saagie sont utilisées pour le développement de modules de traitement.

Les trois lots sont répartis entre les différents partenaires. Dans le cadre lot 1 (voir figure 5.1), Saagie apporte ses compétences en architecture de la donnée et ses compétences en science de la donnée. Saagie apporte également sa plateforme Big Data. La plateforme Big Data permet au projet de partir d'un socle technique prêt à l'emploi et orientée production. Cela permet aux différents acteurs de pouvoir se concentrer sur la conception des algorithmes et sur les réglages avancés des traitements. La plus grande difficulté concerne la parallélisation des traitements. Les réglages avancés de l'architecture sont une grande partie de ses contributions : il s'agit de la préparation de la plateforme pour les particularités des traitements, de l'optimisation de la collecte des données et de l'accompagnement des différents acteurs pour optimiser les traitements. Enfin, Saagie prend en charge l'intégration technique, le nettoyage des données, la gestion du flux temps réel avec des réseaux sociaux comme Twitter, la mise au point du démonstrateur permettant de valider en temps réel les résultats des algorithmes et la capacité du système à prendre en charge un gros volume de données en temps réel. Dans le cadre du lot 2 et 3, Saagie apporte également ses compétences en apprentissage automatique, ainsi qu'à l'élaboration des algorithmes et leurs réglages.

Dans le cadre du lot 2, le LITIS apporte ses compétences en détection d'opinions, de sentiments et d'émotions dans des textes, ainsi qu'en analyse et modélisation du dialogue. En particulier, l'utilisation de contextonymes [38, 135, 136] comme support de reconnaissance du contexte d'utilisation de mots est exploitée dans des méthodes hybrides de détection d'opinions. Airbus est en charge de la détection d'opinion radicale en « arabizi » [4]. Plus précisément, Airbus apporte des technologies linguistiques et statistiques (patrons linguistiques JAPE pour Gate et apprentissage profond) dédiées à l'analyse d'opinions et développées dans le cadre de précédents projets (cf. références). Airbus travaille sur leur portage pour le traitement des alphabets de chat arabe et leur adaptation à la détection de tweets radicaux.

Dans le cadre du lot 3, Airbus contribue à la tâche de détection d'acteurs d'influence en utilisant des algorithmes de typologie de graphe. Cette mesure de l'influence peut être générale ou bien porter sur une thématique précise (un score d'influence possible pour chaque thématique abordée). Le LITIS propose une modélisation à base de systèmes multi-agents afin de reconnaître dans un premier temps un ensemble de com-

4. Alphabet de tchat arabe

portements individuels participant à un groupe d'influence. Dans un second temps, toujours à l'aide d'une approche SMA et de métriques adaptées, l'identification de groupes d'influence sur des topiques identifiés sera réalisée par des approches de détection de communautés dans des graphes. Les premiers résultats obtenus par le Litis sur la détection de communautés et de topiques ont obtenu des retours très positifs de la communauté [39, 40].

La constitution de corpus d'entraînement et de test pour la détection d'opinions sur des tweets en langue arabe et à thématique civile (revue produits) est à la charge du LITIS en recourant à de la sous-traitance afin de diminuer les risques.

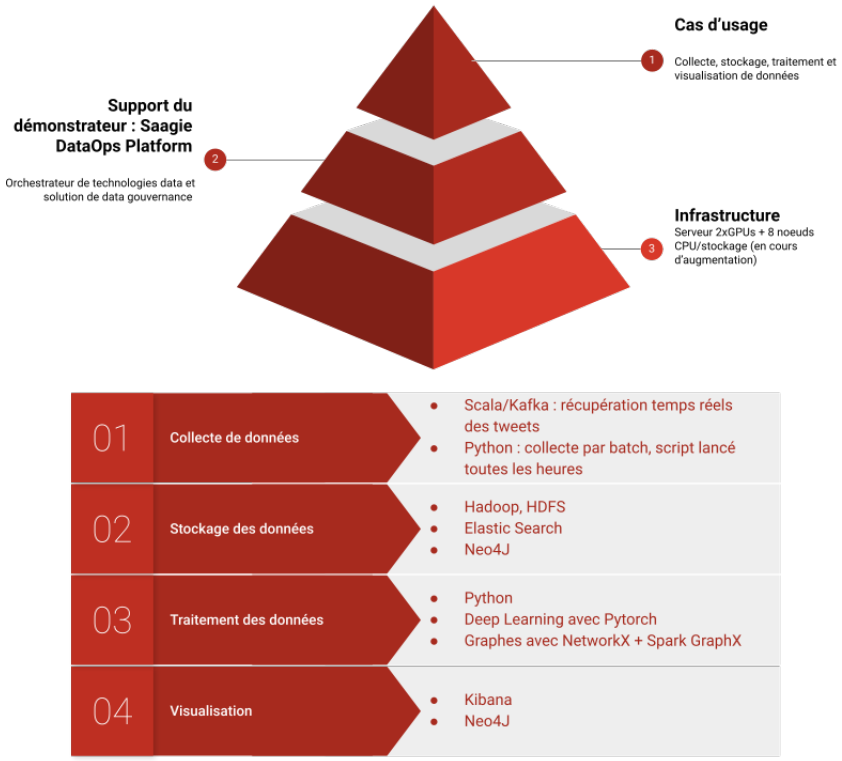

FIGURE 5.1 – Description du lot 1

5.1.3 Jeu de données radicalisation

Malgré une facilité d'accès aux données, collecter un corpus sur la thématique de radicalisation peut être complexe, car les données sont rapidement supprimées des réseaux sociaux. Une partie du budget du projet SAPhIRS est donc consacrée à la collecte et à l'annotation d'un jeu de données. En raison du manque de données disponibles publiquement sur le sujet, nous avons fait appel à la société ELDA [5] pour étiqueter une quantité importante de données Twitter, en respectant un certain nombre de critères. Le corpus collecté doit tout d'abord être annoté en objet et bien évidemment en polarité relativement à l'objet. Il doit également contenir des données en français et anglais, ainsi que de l'arabizi. Enfin, dans le but de pouvoir effectuer de la détection

5. http://www.elda.fr/en/

de changements d'opinion, ce corpus doit être constitué de séquences d'au moins vingt tweets provenant d'un même utilisateur.

La société nous a livré 100 000 tweets annotés avec un objet d'opinion et une polarité (positif, négatif, neutre). L'annotation a été effectuée en deux temps. Un premier échantillon de données annotées nous a permis de clarifier les objectifs de l'étiquetage final. Puis dans un second temps, nous avons pu évaluer le jeu de données définitif. Cependant, cette quantité est toujours très faible pour construire un modèle capable de généraliser sur des millions de tweets. De plus, ce type d'annotation est sujet à des interprétations humaines, de ce fait le jeu de données comporte un biais. Ainsi, nous nous retrouvons dans un contexte où il y a peu de données annotées et des annotations peu fiables.

5.1.4 Jeux de données et modèles pour la communauté

LAD (*Large Arabizi Dataset*)
Une large collecte de données de tweets en arabizi a été réalisée par Airbus avec la librairie de webscraping Twint. Plus précisément, les tweets récupérés sont sur la période 2015 à 2019, et concernent l'arabizi égyptien, grâce à une liste de 48 mots-clés sélectionnés par des experts du langage. Une étape de filtrage a ensuite été effectuée pour supprimer une grande partie de bruit : URL, images... Afin de protéger les données privées, les données collectées ont été anonymisées et toutes les mentions ont été remplacées par NONAME. À l'issue de ces traitements, le jeu de données est constitué de 7,7 millions de tweets en arabizi égyptien. Cela a donné lieu à une publication dans la conférence Coling [3]. Le corpus intitulé LAD pour *Large Arabizi Dataset* est disponible pour la communauté [6].

SALAD (*Sentiment Annotations from LAD*)
À partir du jeu de données LAD présenté précédemment, 1 700 tweets sélectionnés aléatoirement ont été annotés par des experts de ce langage. Ces tweets ont été annotés en 5 catégories :

positif si le tweet exprime un sentiment positif

négatif si le tweet exprime un sentiment négatif

neutre si aucun sentiment n'est présent dans le tweet

conflit si le tweet présente à la fois des sentiments négatifs et positifs

conflit texte/émoticônes si le texte est en contradiction avec les émoticônes présents dans le tweet

La répartition des tweets par classe est présentée dans le tableau 5.1.

5.2 Détection d'influences et de communautés

La première chaîne de traitement réalisée concerne la détection d'influenceurs et de communautés. Ces travaux ont été réalisés par le LITIS et Airbus Defence and Space,

6. http://saphirs.projets.litislab.fr/

Positif	859 (50,5%)
Négatif	422 (24,8%)
Neutre	203 (11,9%)
Conflit	86 (5%)
Conflit texte/émoticônes	130 (7,6%)

TABLE 5.1 – Répartition des tweets pour chacune des classes dans SALAD

et intégrés sur le démonstrateur par Saagie. Dans cette partie, nous décrirons cette chaîne de traitement.

5.2.1 Base de données graphe

Les tweets sont stockés au format CSV sur un système de fichier distribué (HDFS). Ce type de stockage et ce format ne sont pas optimaux pour réaliser des recherches ou des parcours du graphe d'interaction. Ainsi, nous avons intégré une base de données graphe et développé un module d'injection des données CSV dans cette base de données. La base de données graphe choisie est OrientDB, elle permet de requêter le graphe d'interaction via des requêtes dans un format proche du format SQL, et permet de visualiser les résultats sous forme de graphe.

5.2.2 Détecteur d'influenceurs à partir de graphes d'interactions

La détection d'influenceurs est effectuée en plusieurs étapes. Dans un premier temps, le graphe d'interactions, qui relie les utilisateurs ayant eu au moins une communications ensemble, est créé. À partir de ce graphe, les utilisateurs qui communiquent souvent ensemble sont regroupés, ce qui permet d'obtenir plusieurs groupes. Chacun de ces groupes est ensuite étudié afin de détecter le ou les influenceurs qui essaient de manipuler les autres membres. La détection d'influenceurs est également effectuée sur le réseau global afin de connaître les relations entre les différents groupes.

Cette méthode est appliquée sur des données Twitter. Après la création du graphe d'interaction, le graphe est donné en entrée à l'algorithme de détection de communauté de Leiden, afin d'obtenir les groupes dans lesquels les leaders d'opinion tentent d'exercer une influence. Puis l'algorithme Page rank est utilisé afin de détecter les influenceurs globaux et locaux, contenus dans chaque groupe.

5.2.3 Détecteur de communautés d'utilisateurs et d'influenceurs

L'émergence des médias sociaux permet d'étudier la structure sociale des relations humaines et des interactions en ligne, déclenchant des questions sur les opérations d'information et les opérations d'influence. La littérature sur la détection des acteurs clé et des influenceurs est abondante au niveau individuel, elle doit cependant être complétée au niveau communautaire.

En effet, les études en psychologie mettent en évidence la puissance du « groupe » et la variété de ses impacts sur le comportement individuel. En informatique, une étude sur la polarisation des attitudes dans les débats en ligne conclut de la même manière : les groupes peuvent également être considérés comme des influenceurs. L'impact des groupes de comptes coordonnés est devenu un centre d'intérêt pour l'analyse des médias sociaux. Les groupes sont ici considérés comme des communautés thématiques et structurelles : les membres des groupes partagent des centres d'intérêts communs, interagissent et discutent ensemble.

Les travaux de caractérisation des communautés d'utilisateurs ont été poursuivis, en comparant trois jeux de données issus de trois réseaux sociaux différents (Reddit, Twitter et Discord) sur la thématique du projet.

Les communautés ont été détectées à l'aide d'algorithmes différents, dont Label Propagation, Oslom, Link et Louvain : ces méthodes mettent l'accent sur l'optimisation de mesures topologiques différentes, résultant en une variété de types de communautés. Par exemple, Link a tendance à identifier des petits groupes très densément liés, tandis que Louvain agrège de nombreux nœuds faiblement connectés. Nous avons amélioré les mesures de cohésion thématique, principalement en modélisant les thématiques différemment (de manière continue, à l'aide des *embeddings* des documents, plutôt que par un label comme précédemment). Ces travaux ont donné lieu à une publication lors de Web Intelligence 2019[37].

5.3 Détection d'opinions et de haine

En parallèle à la partie dédiée à la détection d'influences et de communautés, différents modules de détection d'opinions et de haine ont été mis en place. Les modules décrits dans cette partie ont été développés par Saagie. Nous présentons chacun d'entre eux, ainsi que leur fonctionnement.

5.3.1 Collecte de tweets et stockage

La collecte des données est la première étape effectuée. Elle est effectuée grâce à l'API Twitter officielle, limitée à 1 000 requêtes toutes les 24h. Pour contourner cette limitation, l'API officielle a été combinée à la librairie de webscraping Twint. L'objectif de cette collecte est de récupérer de gros volumes de données (environ 12 millions de tweets collectés pour le projet), qui vont ensuite permettre d'entraîner les différents modules. Ces données serviront par exemple à rendre la traduction plus performante, ou encore à permettre une meilleure classification des tweets radicaux.

Les tweets collectés sont stockés sur un système de fichiers distribué (HDFS) par l'intermédiaire de la solution de Saagie, afin d'être facilement exploitables pour l'analyse et la visualisation. Deux modules ont été développés pour collecter les tweets voués à être analysés (voir schéma en figure 5.3). Un premier module collecte en temps réel des tweets en rapport avec des mots-clés choisis par l'utilisateur. Il utilise l'API Twitter officielle. Un second module est lancé manuellement pour collecter des tweets sur des utilisateurs ciblés. Il permet de récupérer l'ensemble de ses tweets sur une pé-

[#franceattack], [Boycott France], [#boycottfrenchproducts], [#BoycottFrenchProductsForever], [مقاطعه_المنتجات_الفرنسيه], [#we_love_mohammad_ﷺ_challenge], [#,[الا_رسول_الله], [#إلا_حبيب_الله], [#مقاطعة_المنتجات_الفرنسية], [#ماكرون_يسئ_للنبى], [#رسولنا_خط_احمر], [#حبيبى_يا_رسول_الله], [#]], [#Sauf_le_messager_de_Dieu], [#Sauf_le_bien_aimé_d_Allah], [#Notre_religion_est_la_religion_droite_de_la_vérité], [مقاطعه_المنتجات_الفرنسيه], [#boycottfrance], [#La_plus_grande_création_de_Dieu_Muhammad1], [#sauf_le_messager_de_dieu1], [#Sauf_le_bien_aimé_d_Allah1], [#Notre_religion_est_la_religion_droite_de_la_vérité1], [#Boycottez_les_produits_français1], [#,[دیننا_دین_الحق 1], [#إلا_حبيب_الله 1], [#الا_رسول_الله 1], [#محاكمه_يوسف_هاني]], [we_love_mohammad_challeng], [مقاطعه_المنتجات_الفرنسيه], [#Sauf_le_messager_de_Dieu1], [#Sauf_le_bien_aimé_d_Allah1], [#Sauf_le_messager_de_Dieu], [#Sauf_le_bien_aimé_d_Allah], [#La_plus_grande_création_de_Dieu_Muhammad], [#Boycott_des produits_ Français], [#,[الا_حبيب_الله], [#الا_رسول_الله], [#اللَّهُمَّ_صَلِّ_عَلَى_مُحَمَّدٍ_وَّعَلَى_آلِ_مُحَمَّدٍ], [#مقاطعة_المنتجات_الفر,"#ماكرون_يسئ_للنبى], [#رسولنا_خط_احمر], [#حبيبي_يا_رسول_الله]"#La_ilaha_illallahu_muhammadur_rasulullah], [#Stop_Islamophobia_in_French], [#Boycott_French_Products], [#We_Follow_Muhammad_SW], [#we_love_mohammad_ﷺ], [#MyProphetMyHonourﷺ], [#boycottfrenchproducts], [#WeHateFrancegovernment], [#BoycottFrench], [#BoycottFrance], [#Boycott_France], [#Boycott_French_Products], [#WeHateFrancegovernment], [#BoycottFranceProducts]

FIGURE 5.2 – Liste des mots-clés utilisés pour la collecte sur la thématique du boycott des produits français

riode choisie. La librairie de scraping Twint est utilisée par ce dernier. L'ensemble des tweets est récolté au format JSON, puis converti en CSV pour faciliter leur traitement ultérieur.

Une collecte à l'initiative de la DGA et mise en œuvre par Saagie a été effectuée sur la thématique du boycott des produits français sur une période d'un mois avant la fin du projet. Ceci a permis de récolter plus d'un million de tweets. Ces données ont servi pour la démonstration finale en condition réelle. La liste des mots-clés utilisés est présentée en figure 5.2.

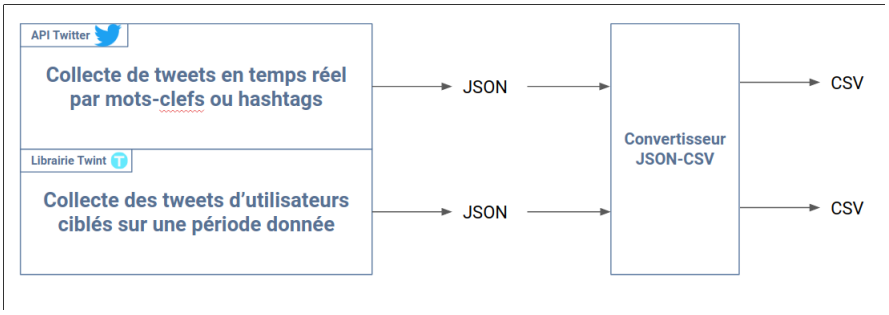

FIGURE 5.3 – Schéma de la collecte de tweets via l'API officielle et Twint

5.3.2 Détection de langue

La première étape de la chaîne de traitement des tweets collectés consiste à détecter la langue de chacun des tweets (voir figure 5.4). En effet, les tweets collectés par les modules de collecte peuvent être de toutes langues. Afin d'adapter les traitements à réaliser pour chaque langue, un module de détection de langue a été créé.

Pour cela, 21 langues à détecter ont tout d'abord été identifiées, dont l'arabizi, puis

100 000 tweets ont été collectés pour chacune des langues. La collecte a été réalisée à l'aide du mot-clé spécifique « mais » dans chaque langue (ie. « but » en anglais, « mais » en français, « pero » en espagnol, etc). Ensuite, un réseau de neurones profond a été entraîné sur cette tâche de classification de langage en affinant un modèle BERT [33]. Ce modèle est capable d'identifier la langue avec un score F1 macro de 92% sur le jeu de test.

Il faut noter que les jeux de données d'apprentissage et de test sont bruités, certains tweets ont un mauvais label. En effet, en se basant uniquement sur la présence d'un mot-clé dans le tweet pour l'annoter dans une certaine langue, certains seront mal étiquetés, puisqu'une même séquence de caractères peut exister dans plusieurs langues en ayant des sens complètement différents.

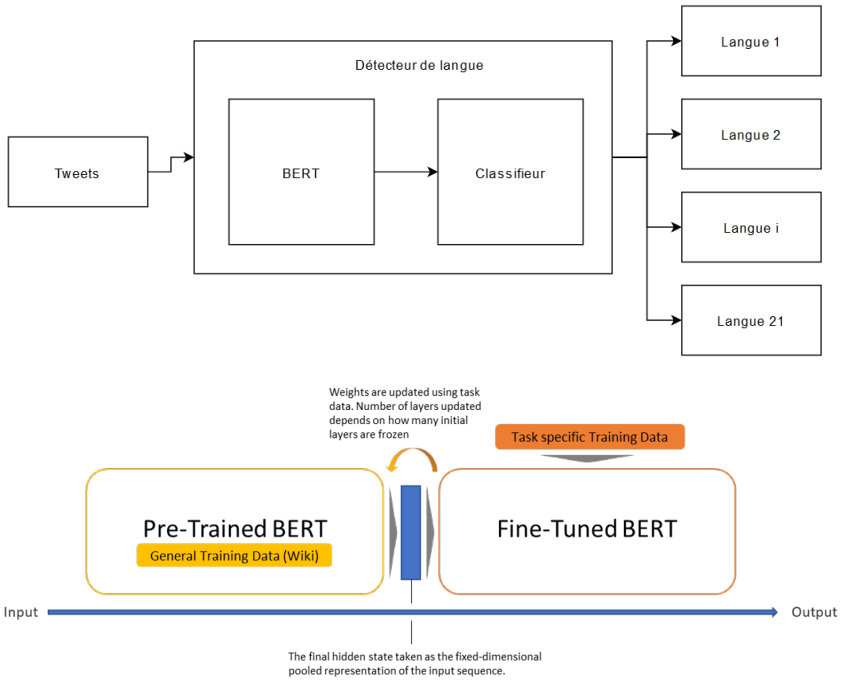

FIGURE 5.4 – Schéma de la détection de langue

5.3.3 Détection de dialecte

Une fois la langue détectée, si le tweet est en arabe, il peut être intéressant de connaître le dialecte utilisé. En utilisant le jeu de données MADAR6[7], un modèle a été entraîné afin de développer un module de détecteur de dialecte. Ce détecteur permet de classifier les tweets en arabe parmi 6 classes :

- MSA (Modern Standard Arabic)
- ainsi que 5 dialectes différents
  - Golf d'Aden
  - Golf
  - Levant

7. https://camel.abudhabi.nyu.edu/madar-shared-task-2019/

— Bassin du Nil

— Maghreb

Ce module, illustré en figure 5.5 se place juste après le détecteur de langue, et uniquement pour les tweets qui ont été classifiés en langue arabe.

Le classifieur de dialecte s'appuie sur le modèle générique pré-appris DistilBERT [131]. Ce modèle a été adapté grâce à du *fine-tuning* sur la tâche spécifique de détection de dialecte. Le modèle affiné sur ces données est capable d'identifier le dialecte avec un score F1 macro de 88,1% sur le jeu de test.

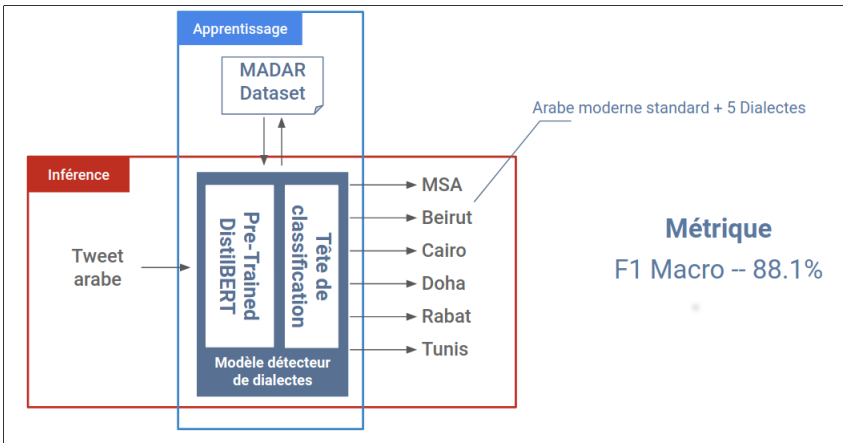

FIGURE 5.5 – Schéma de classifieur de dialecte

5.3.4 Translittération

L'arabizi, qui correspond à l'alphabet de chat arabe est peu traité en TAL. Les modèles entraînés sur cette langue sont donc rares ou inexistant. Pour pallier cette difficulté, un module de translittération a été réalisé pour transformer les tweets en arabizi vers la langue arabe moderne, comme dans la figure 5.6.

Pour ce faire, trois méthodes sont proposées. La première utilise l'API Google Translate, qui apparaît comme la plus fonctionnelle d'après des experts arabophones. Cependant, le nombre de requêtes autorisées par cette API est limité par jour et peut s'avérer problématique si de nombreux textes en arabizi doivent être translittérés dans une courte période de temps. Aussi, l'API Yamli, qui est très employée par la communauté, a été utilisée. Son inconvénient principal réside dans le temps de traitement des requêtes. Enfin, pour la dernière méthode, il s'agit d'un système développé en Python, en se basant sur des règles expertes qui couvrent la majorité des cas de figure.

Ainsi, il est possible de choisir le module adapté en fonction des contraintes de la tâche et de problématiques de souveraineté.

5.3.5 Traduction

Puisque l'anglais reste la langue la plus traitée en TAL, il peut être judicieux de convertir le texte rédigé en langue « rare » afin de faciliter le traitement.

FIGURE 5.6 – Exemple de translittération

Ainsi, un module de traduction a été intégré à la chaîne de traitement. Ce dernier permet de traduire des textes en langue arabe moderne vers la langue anglaise (voir figure 5.7). Ce choix a été fait, car il s'avère que le module de détection de polarité est plus efficace après traduction en langue anglaise que directement en langue arabe. Cela pourrait s'expliquer par une meilleure généralisation des modèles en langue anglaise due à une plus grande quantité de données lors de leur pré-apprentissage.

Le module de traduction arabe-anglais s'appuie sur un modèle à disposition au sein de la bibliothèque *open source* HuggingFace.

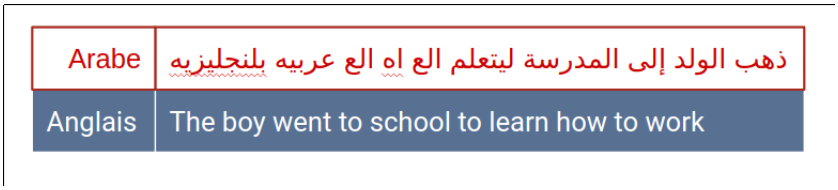

FIGURE 5.7 – Exemple de traduction

5.3.6 Détection de haine

La haine peut être considérée comme une composante de la radicalisation. Pour cette raison, l'ajout d'un classifieur de haine à la chaîne de traitement a été jugé utile, afin d'améliorer la classification de radicalisation.

Dans un premier temps, Saagie a voulu développer un module de détection de discours de haine en reprenant puis affinant les poids du modèle BERT-base publié sur HuggingFace. Les jeux de données français et arabizi étant rares ou inexistants, l'entreprise s'est tout d'abord limitée à l'anglais, et a utilisé HatEval 2019 et OffensEval 2019, deux jeux de données issus de la compétition SemEval, à laquelle elle a participé. Ceci nous a permis de mettre en place une première version de classifieur de discours haineux pour la langue anglaise. Le modèle résultant est un classifieur binaire qui détermine si un tweet anglais est haineux ou non haineux. Une probabilité est retournée selon la classe renvoyée. Après avoir testé le modèle sur les données de validation, nous avons obtenu un score F1 de 80,66.

En 2020, plusieurs modèles spécialisés dans la détection de haine en différentes langues ont été publiés sur HuggingFace : les modèles Hate-ALERT [2] fondé sur l'architecture de BERT. Ces modèles ont été entraînés sur des données de haines dans différentes langues. Vingt-cinq modèles étaient disponibles. Ces nouveaux modèles ont permis d'obtenir de meilleures performances sur l'anglais, et d'atteindre un score F1 de 82,35. Au vu des résultats présentés dans le tableau 5.2, le choix de modèle à intégrer au

pipeline s'est évidemment porté sur le modèle Hate-ALERT pour les tweets anglais. En effet, ce modèle ayant été entraîné sur une quantité plus importante de tweets, il obtient donc de meilleures performances (+1,69 points de score F1)

Pour les langues français et arabe, aucune donnée d'apprentissage n'était disponible. Le choix a donc été fait de s'appuyer totalement sur les modèles Hate-ALERT français et arabe publiés sur HuggingFace. Ces deux modèles ont donc également été intégrés au pipeline de détection de radicalisation, ce qui a permis d'avoir un détecteur de haine par langue traitée.

Modèle	Score F1
BERT affiné	80,66%
Hate-ALERT English	82,35%

TABLE 5.2 – Résultats obtenus pour la détection de haine

5.3.7 Détection d'objet d'opinion

Afin de connaître l'objet des tweets, et de permettre de détecter la polarité de l'opinion envers cet objet, un détecteur d'objet d'opinion a été mis en place.

L'objectif de ce détecteur d'objet est d'identifier l'objet contenu dans chaque tweet. Il est basé sur le jeu de données recueilli et annoté par la société ELDA pour le projet. À la base, ce jeu de données contient 100 000 tweets annotés en 1 115 catégories différentes. Dû à une très grande disparité entre le nombre de tweets par catégorie (86% des catégories contient moins de 50 tweets), un regroupement des catégories en un nombre beaucoup plus restreint a été réalisé au moyen d'un clustering DBSCAN sur leur représentation vectorielle.

Trois regroupements différents ont été comparés. Le premier contient les catégories avec plus de 50 tweets. Les deux autres regroupements se basent sur une validation manuelle. Pour chaque configuration, un modèle DistilBERT a été affiné afin d'être capable de classifier les tweets selon la liste des catégories. Le tableau 5.4 montre les scores pour ces trois configurations.

Le modèle choisi et intégré au *pipeline* correspond au modèle capable de classifier les tweets en treize classes différentes. La liste de ces différents objet est présentée dans le tableau 5.3.

'Anti-Haine', 'Anti-terrorisme', 'Anti-Religion', 'Armes', 'Sports', 'Événements', 'Haine', 'Incompréhensible', 'Media', 'ONG', 'Organisation violente', 'Pays/Villes', 'Politique', 'Religion'

TABLE 5.3 – Liste des treize catégories sélectionnées pour la détection d'objet

5.3.8 Détection de polarité d'opinion

Après avoir détecté l'objet de l'opinion, il faut maintenant classifier la polarité de celle-ci. Pour cela, un module de détection de polarité a été créé. Ce module s'appuie

Classes regroupées	Modèle	Score F1 Macro
13	DistilBERT + 2 layers attention	52%
18		49%
163		31%

TABLE 5.4 – Scores F1 pour les différents nombres d'objets testés

sur des modèles statistiques entraînés sur le jeu de données recueilli et annoté par la société ELDA. Les 100 000 tweets du jeu de données sont annotés en trois classes : négatif, positif, neutre. Ces modèles se basent sur le *fine-tuning* de différents modèles de langue pré-entraînés sur des corpus génériques conséquents.

Pour chacune des langues, les modèles ci-dessous ont été affinés sur les tweets de la langue correspondante dans le corpus annoté par ELDA :

Français : utilisation du modèle CamemBERT

Anglais : utilisation des modèles BertTheseus et BertMultiLang, combinaison des prédictions

Moderne Arabe : utilisation du modèle BertMultiLang

Arabizi : utilisation des modèles BertTheseus et BertMultiLang, combinaison des prédictions.

Le tableau 5.5 présente les métriques de validation des prédictions sur le jeu de test pour chaque langue.

Langue	Modèle	Score F1
Français	CamemBERT	74%
Anglais	BertTheseus + BertMultiLang	76%
Moderne Arabe	BertMultiLang	79%
Arabizi	BertTheseus + BertMultiLang	80%

TABLE 5.5 – Scores F1 de la détection de polarité d'opinion sur les données du corpus annoté par ELDA

5.4 *Pipeline*, déploiement et visualisation

Chacun des modules décrits précédemment est organisé sous forme de pipeline dans le démonstrateur Saagie. Dans cette partie, nous verrons comment chacun de ces éléments s'organise entre eux afin de former le projet final livré à la DGA.

5.4.1 Démonstrateur Saagie

Le démonstrateur qui a été développé s'appuie sur la solution Saagie, qui est un orchestrateur DataOps. Le DataOps est une méthodologie collaborative de gestion des données dont l'objectif est d'améliorer la communication, l'intégration et l'automatisation des flux de données entre les gestionnaires et consommateurs de données au sein

d'une organisation. La solution permet d'accélérer le développement de projets Data et leur mise en production. Elle fournit un ensemble d'outils et un cadre pour adopter une approche inspirée des pratiques DevOps pour le monde de la donnée.

La solution s'appuie sur des containers (Docker) pour permettre à tout utilisateur de créer et déployer des technologies/outils qui ne seraient potentiellement pas présentes de base. Une gestion fine des droits permet en outre de gérer l'accès aux données et aux projets par utilisateur et groupe.

La solution Saagie a été développée de telle sorte qu'elle ne verrouille pas l'utilisateur en son sein. Les développements effectués peuvent être utilisés en dehors du produit sans difficulté grâce à la conception modulaire via containers et grâce à l'utilisation de technologies issues de l'open source (Python, R, Spark, Scala…).

Tous les développements effectués dans le cadre de ce projet sont réalisés selon une approche modulaire. Chaque module peut être exécuté indépendamment des autres, en respectant les formats d'entrées définis au sein de chaque module. Ces modules peuvent être exécutés en dehors de la solution.

Plusieurs alternatives sont possibles pour réaliser et déployer un module sur la solution :

— L'écriture d'un script dans le langage souhaité (Python, R, etc) et le déploiement sous forme de job via les capsules respectives

— La création et le déploiement d'une image Docker intégrant des librairies et technologies requises au bon fonctionnement d'un script : les images Docker sont déposées sur un *repository* Docker (e.g. Dockerhub) afin d'être ensuite déployées dans la solution.

Les modules sont regroupés dans une logique de pipeline via un outil associé dans le produit. Ce dernier permet d'exécuter les modules les uns après les autres et de suivre l'avancement des traitements.

La logique de modularité permet d'être agile et de s'adapter continuellement à l'évolution des technologies et connaissances, en offrant la possibilité de remplacer chaque module unitairement ou d'en ajouter au sein des différentes chaînes de traitement (pipeline).

Aussi, il est important de savoir que les différentes chaînes de traitements se font en parallèle.

5.4.2 *Pipeline* de modules

Le premier pipeline du projet est chargé de la tâche de détection de discours de haine et d'opinions. Le schéma en figure 5.9 montre l'enchaînement des différents traitements de ce pipeline. Dans un premier temps, des tweets sont récupérés, soit en temps réel sur des mots-clés spécifiques, soit par lots pour cibler des utilisateurs. Ils sont ensuite convertis en fichier plat (CSV) et envoyés à un détecteur de langue. Si la langue du tweet est en arabizi, un module de translittération se charge de le convertir en arabe.

Puis un détecteur de haine par langue (Français, Anglais, Arabe) classifie les tweets. En parallèle de ces traitements, une détection d'objet d'opinion est réalisée sur tous les tweets, puis, après traduction en anglais des tweets en arabe, l'ensemble des tweets est classifié par des modèles de détection de polarité (différents modèles par langue). Enfin, un traitement parallèle est réalisé pour détecter le dialecte des tweets en arabe. L'ensemble des résultats de chaque traitement est stocké au fur et à mesure dans le lac de données.

Le second pipeline implémenté concerne la détection des leaders d'influences et des communautés par analyse de graphe. Ce pipeline fonctionne en parallèle du pipeline de détection d'opinion et de haine et s'appuie sur les mêmes données en entrées. Elles proviennent donc du collecteur de tweets temps réel via l'API Twitter officielle et du collecteur de données antérieures sur des utilisateurs ciblés via la librairie Twint. Ces données sont injectées dans une base de données graphe, puis envoyées à deux modules parallèles. Le premier est chargé de détecter les communautés dans le graphe d'interactions. Le second réalise la détection des influenceurs dans ce même graphe. Tous les résultats des traitements sont stockés dans le lac de données.

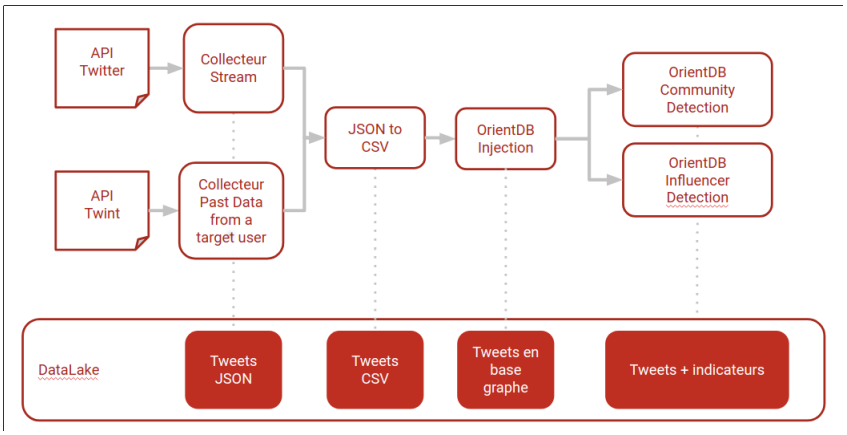

FIGURE 5.8 – Pipeline de détection d'influences et de communautés

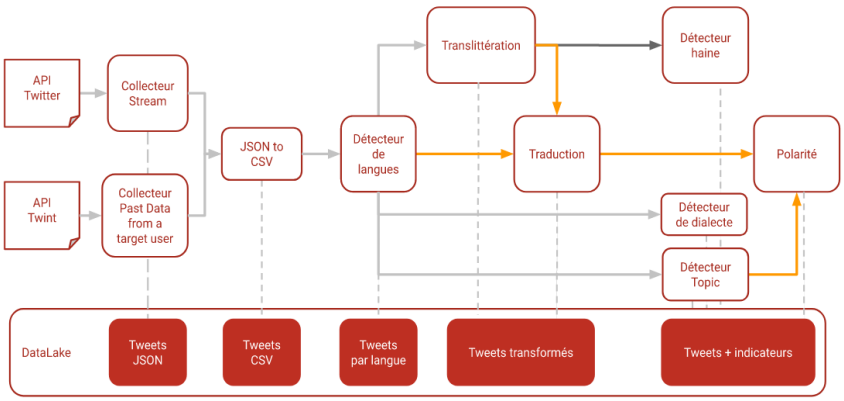

FIGURE 5.9 – Pipeline de détection de radicalisation

5.4.3 Stockage des données et déploiement du *pipeline*

Chaque module développé est associé à un répertoire dans le système de fichier distribué. Ce répertoire contient un dossier « stream/ » qui reçoit les fichiers à traiter, et un dossier « history/ » qui reçoit les fichiers traités.

Chaque modèle d'apprentissage profond est déployé au sein d'une instance de TorchServe qui permet d'exposer un modèle et de le requêter via API REST.

Un script appelé *Pipeline Block* est chargé de récupérer les données, requêter le modèle d'apprentissage profond associé et faire le lien avec les autres traitements du pipeline.

Ce script fonctionne comme suit. Il charge les fichiers présents dans le dossier « stream/ » par micro-batch, ces derniers sont envoyés par requêtes API REST à Torchserve pour en obtenir des prédictions d'un modèle cible. Enfin, il écrit les résultats dans le répertoire « history/ » et dans le répertoire « stream/ » des traitements qui suivent dans le pipeline.

Des conditions peuvent être ajoutées pour sélectionner des tweets et des colonnes en particulier lors de la demande de prédiction.

Les prédictions sont ajoutées en colonnes supplémentaires dans les CSV des données initiales. Le schéma en figure 5.10 résume le fonctionnement du script *Pipeline block*.

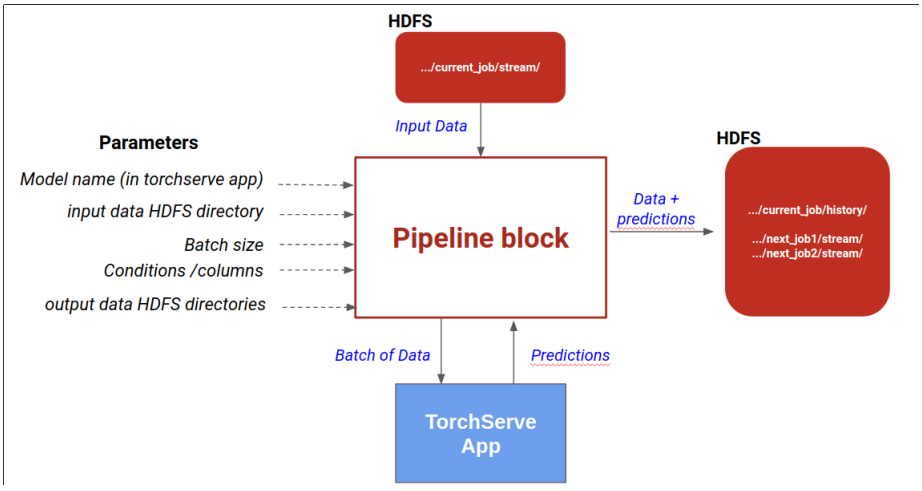

FIGURE 5.10 – Fonctionnement des bloc de pipeline

5.4.4 Visualisation des résultats

L'ensemble des traitements réalisés dans les pipelines fournit un ensemble d'indicateurs pour caractériser les messages échangés, les utilisateurs, les influenceurs, les communautés.

Pour pouvoir explorer, filtrer, identifier des utilisateurs, les groupes, des tableaux de

bords interactifs ont été créé via l'outil PowerBI, qui permettent d'interagir avec les données et de les visualiser. Les figures suivantes présentent les différents tableaux de bord réalisés.

Le premier tableau de bord (figure 5.11) donne des statistiques globales sur l'ensemble des utilisateurs comme la répartition des messages haineux, la polarité de tweets, les sujets abordés et les langues. Ils peuvent être filtrés selon diverses caractéristiques.

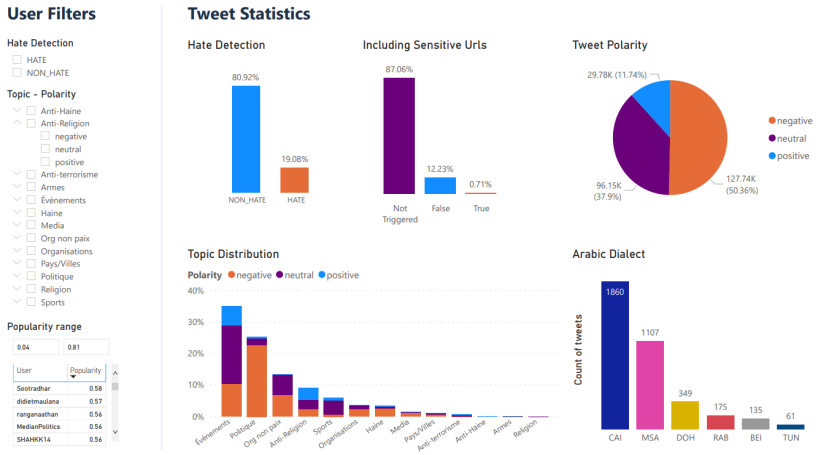

FIGURE 5.11 – Visualisation de statistiques globales

Ce deuxième tableau de bord (figure 5.12) permet d'avoir une visualisation du point de vue d'un utilisateur sélectionné. Cela permet d'avoir accès aux mots-clés qu'il a employé, à ses statistiques utilisateur comme sa langue majoritaire, son nombre de *followers*, etc., ainsi qu'à ses messages.

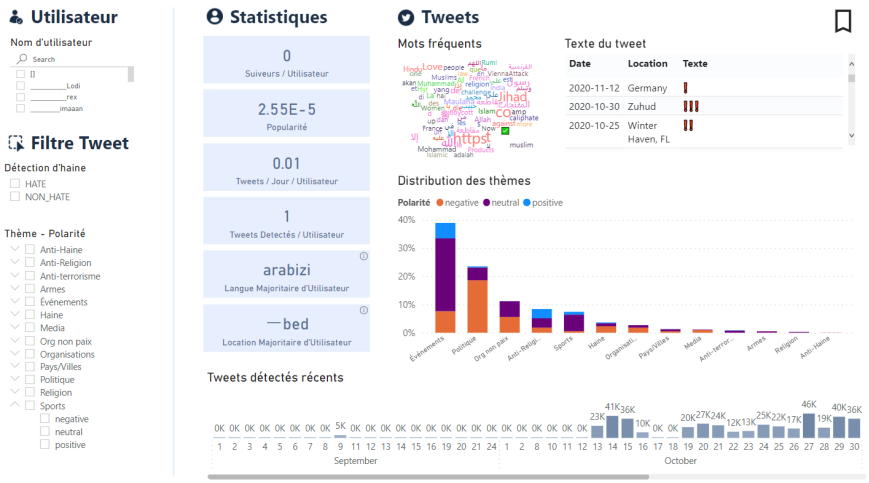

FIGURE 5.12 – Visualisation du point de vu d'un utilisateur

Ce troisième tableau de bord (figure 5.13) sert à la recherche de tweets. Il permet de filtrer les tweets en saisissant certains mots-clés, ainsi que sélectionner les utilisateurs, le vocabulaire et les communautés d'utilisateurs associées.

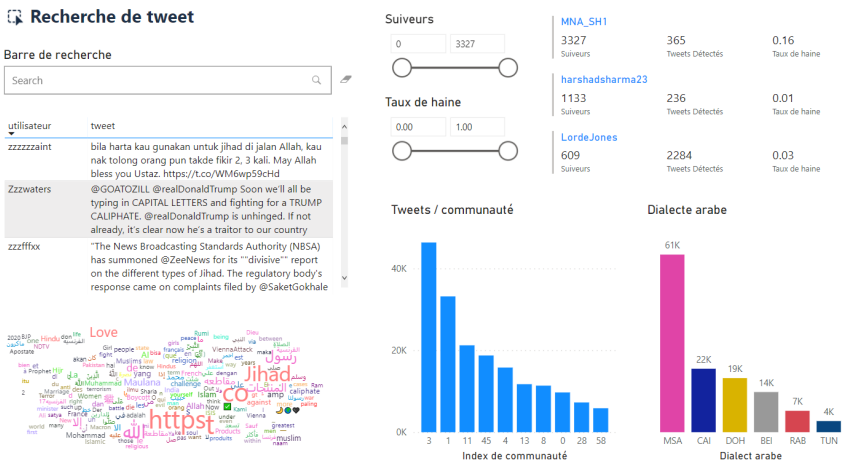

FIGURE 5.13 – Tableau de bord de recherche de tweet

Ce quatrième tableau de bord (figure 5.14) permet de filtrer des communautés et des utilisateurs avec les métriques d'influence.

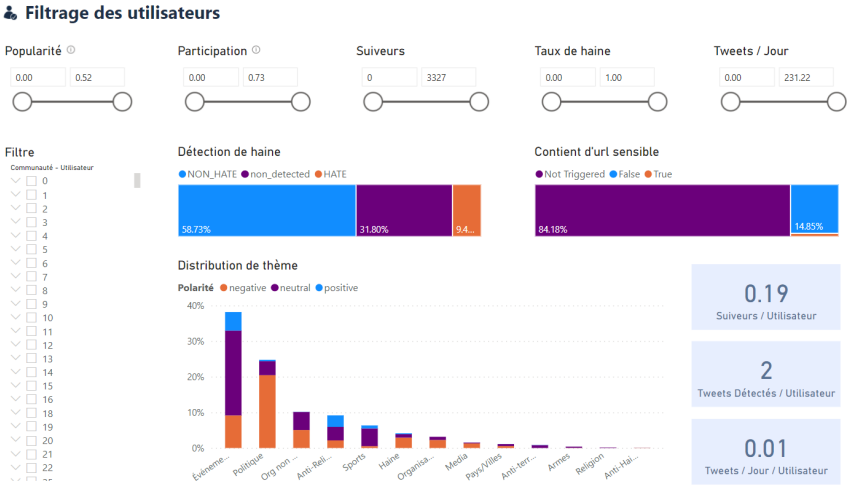

FIGURE 5.14 – Tableau de bord de filtre des utilisateurs et de communautés

Enfin, le cinquième et dernier tableau de bord (figure 5.15) permet d'effectuer la détection de communauté. Il montre des informations sur les communautés qui ont été sélectionnées : les utilisateurs associés, la topologie de la communauté, les mots fréquents et les tweets. Il permet également de filtrer par la langue habituelle de l'utilisateur.

5.5 Difficultés et limite du projet

Plusieurs pistes ont été explorées et abandonnées lors des trois ans de projet. En effet, il était question d'être capable de détecter des changements d'opinion durant ce projet. Cependant, nous n'avions pas anticipé la difficulté à collecter et identifier des changements d'opinion au cours du temps sur des sujets ciblés. Il n'est pas fréquent de trouver ce type de comportement sur Twitter. La société ELDA chargée de la collecte

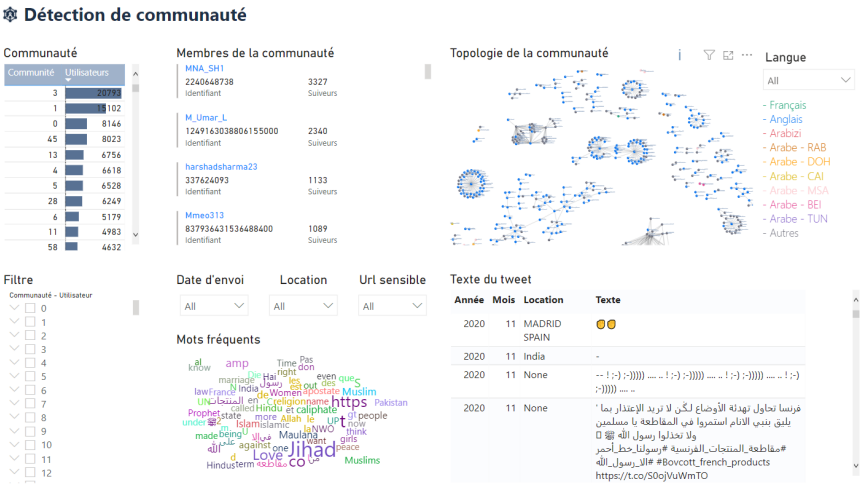

FIGURE 5.15 – Visualisation de la détection de communautés

et de l'annotation n'a pas été en mesure de nous fournir ces données en quantité suffisante.

Un autre sujet qui nous a posé des difficultés est le traitement de l'arabizi. Ce langage n'est pas normé et il n'existe que très peu de jeux de données annotés ou de textes parallèles qui nous auraient permis de créer un modèle de traduction supervisé. Nous avons néanmoins essayé des approches non supervisées pour apprendre un modèle de traduction à partir de textes non annotés. Les résultats se sont avérés très décevants et nous avons privilégié une approche par translittération vers l'arabe puis traduction vers l'anglais. Cette approche n'est potentiellement pas la plus optimale, car des erreurs peuvent à la fois être présentes dans la phase de translittération et dans la phase de traduction.

Nous avions en outre minimisé la difficulté de la tâche d'annotation du jeu de données de la société ELDA. En effet, le nombre de catégories d'objet d'opinion n'était pas limité initialement et nous avons été confronté à un grand nombre de catégories similaires ou contenant un nombre de tweets très faibles. Cela nous a demandé des tâches de pré-traitement pour réduire et regrouper ces catégories.

5.6 Participation à SemEval 2019

Au lancement du projet, le jeu de données de radicalisation constitué par la société ELDA n'était pas encore disponible. Les travaux ont tout de même commencé sur des données proches mais différentes. Pour cela, Saagie a souhaité participer à la compétition SemEval 2019.

Depuis 1998, SemEval est une série de compétitions permettant l'évaluation de systèmes d'analyse sémantique, suivie par une conférence annuelle. Parmi les tâches proposées en 2019, la tâche 5 a semblé intéressante car proche du sujet de cette thèse et du projet SAPhIRS. Cette tâche, « HatEval » [9] correspond à de la détection de haine envers les femmes et les migrants mexicains. Elle fut divisée en deux sous-tâches.

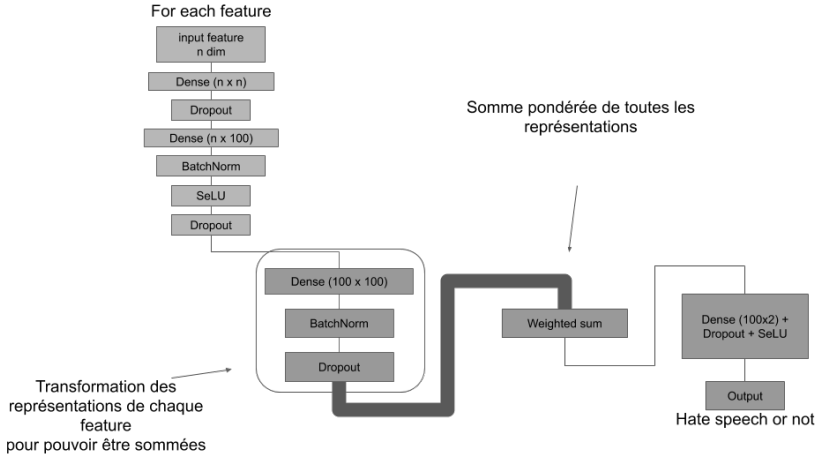

FIGURE 5.16 – Somme pondérée et combinaison de caractéristiques

La première sous-tâche consistait à classifier des tweets comme étant haineux ou non. Puis, dans la seconde sous-tâche, il s'agissait pour les tweets haineux de déterminer s'ils étaient agressifs ou non, et si la cible était générique ou un individu. La tâche était également disponible en deux langues : anglais et espagnol.

Pour chaque langue, un corpus d'entraînement, un corpus de développement et un corpus de test étaient disponibles [9]. Ils contenaient respectivement 9 000, 1 000 et 2 970 tweets pour l'anglais, et 4 500, 500 et 1 599 tweets pour l'espagnol. L'annotation a été effectuée grâce à la plateforme de crowdsourcing Figure 8 [8]. Ces jeux de données étaient assez équilibrés, avec environ 42% de tweets haineux (voir Chapitre 2 pour une description plus détaillée du jeu de données).

Dans le cadre du projet SAPhIRS, le pôle Recherche de Saagie a choisi de participer à cette compétition pour les deux sous-tâches et les deux langues. La 12ème position a été atteinte, sur 68 participants pour la première sous-tâche en anglais, et la 10ème position sur 40 pour l'espagnol. Cette compétition a fait l'objet d'une publication [11]. Dans cette section, nous détaillons la participation à cette compétition. Nous commençons par décrire le modèle, avant de préciser, pour chacune des sous-tâches, les caractéristiques et approches qui ont servi à la détection de haine.

5.6.1 Description du modèle

La faible quantité de données disponibles n'est pas suffisante pour établir une représentation des données satisfaisante. Il faut recourir à un modèle générique pré-entraîné, comme nous avons pu le voir dans le Chapitre 3. Cependant, le domaine de la haine étant très spécifique, un modèle générique ne peut pas être utilisé sans être adapté.

La tendance actuelle en TAL est d'effectuer de l'apprentissage par transfert, et plus particulièrement du *fine-tuning* à partir d'*embedding* ou de modèle de langage, afin de

8. https://www.figure-eight.com/

les adapter aux données et de faire de la classification de texte. Ainsi, il est possible d'utiliser les connaissances d'un domaine source et de les appliquer à un autre domaine cible [128]. C'est ce qui a été fait pour cette tâche en exploitant les modèles de langages faisant l'état de l'art : BERT [33] et GPT [118], présentés en Chapitre 2.

En effet, depuis leur publication, les modèles Transformer sont à l'état de l'art. Lors de la participation à la compétition SemEval 2019, les modèles BERT et GPT étaient disponibles depuis peu. Ces deux modèles se fondent sur le BPE [41], un type de représentation fondé sur les segments de mots. Ces modèles pré-entraînés ont donc été utilisés. Cependant, pour ne pas se limiter à une unique représentation de phrase, une approche de *meta-embedding* dynamique [72] a également été employée, permettant de combiner plusieurs représentations. D'autres caractéristiques ont également été créées, inspirées de [110], et seront détaillées dans la partie suivante.

Le modèle final utilisé est présenté en figure 5.16. Nous pouvons voir que chacune des caractéristiques, aussi bien celles des modèles BERT et GPT que celles crées manuellement, sont combinées entre elles au moyen d'une somme pondéré inspirée du *meta-embedding* dynamique.

5.6.1.1 Autres caractéristiques

Si BERT et GPT ont servi de représentations principales pour le texte, ces modèles ont été combinés à d'autres caractéristiques couramment utilisées en TAL. En effet, s'inspirant de (Pamungkas et al., 2018) [110], différentes caractéristiques ont été créées afin de les fusionner avec les modèles de langage :

Language Model Perplexity Score de perplexité de chaque tweet selon le modèle de langage kenlm [9]

BayesianEncodingHashtag probabilité d'un hashtag selon son label

hashtagUrlPresence Présence d'URL ou de hashtag dans le tweet

Abreviation Décompte du nombre d'abréviation

BagOfPOSTagging Décompte des étiquettes morpho-syntaxiques

NMF *Non-negative Matrix Factorization*, factorisation par matrices non négatives

LDA *Latent Dirichlet Allocation*

BagOfEmojiFeatures Présence d'emojis dans le tweet

nbWords Nombre de mots dans chaque tweet, normalisé par la moyenne

Textstat Lisibilité du tweet [10]

nbChar Nombre de caractères de chaque tweet, normalisé par la moyenne

Ces différentes caractéristiques ont été combinées entre elles, ainsi qu'avec les représentations de BERT et GPT, en utilisant une approche semblable au *meta-embedding* dynamique, qui sera décrit dans la partie suivante.

9. https://github.com/kpu/kenlm
10. https://github.com/shivam5992/textstat

5.6.1.2 *Meta-embedding* dynamique

Après avoir créé les caractéristiques nécessaires, il a fallu les fusionner avec les représentations de BERT et GPT. Parmi les méthodes présentées en Chapitre 4, le *meta-embedding* dynamique [72] a servi de fondement. En effet, le *meta-embedding* dynamique permet de créer un ensemble de différentes représentations, et de laisser le modèle choisir celle qui correspond le mieux en fonction des performances de chacune. Dans la figure 5.17, nous pouvons voir le fonctionnement de cette méthode de fusion.

Afin de combiner les représentations de BERT et GPT ainsi que les différentes caractéristiques créées manuellement, c'est donc le *meta-embedding* dynamique qui a été sélectionné. Cependant plutôt que de permettre au modèle d'apprendre à choisir la représentation la plus pertinente pour classifier les tweets, chacune des représentations a été combinée en utilisant une somme pondérée. C'est cette pondération qui est apprise par le modèle. Ainsi, l'importance de chacune des caractéristiques est déterminée automatiquement. De plus, cette pondération agit comme un sélecteur de caractéristique, en permettant de filtrer celles dont la pondération est au plus bas, et qui n'apporte rien au modèle. Ceci nous a donc permis de sélectionner uniquement les meilleures caractéristiques pour chaque sous-tâche et chaque langue, ce qui sera détaillé plus tard dans cette section.

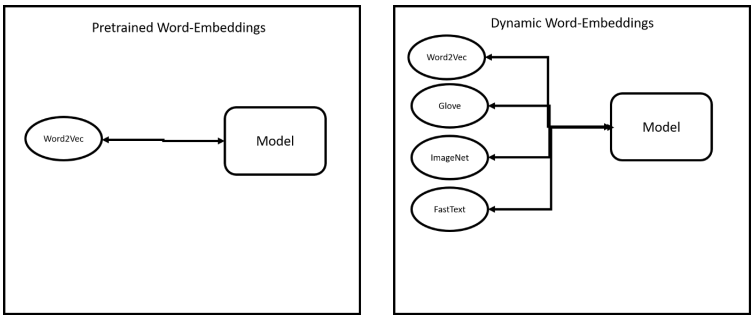

FIGURE 5.17 – Fonctionnement du *meta-embedding* dynamique, schéma extrait de *Dynamic Meta-Embedding : An approach to select the correct embedding*

5.6.1.3 Pré-traitements

Comme nous avons pu le voir en Chapitre 2, traiter des données Twitter nécessite une étape de normalisation, due au vocabulaire spécifique des réseaux sociaux. Pour cette compétition, peu de pré-traitements sur les tweets ont été appliqués, en suivant les recommandations de la littérature (voir Chapitre 2 et 4 pour plus de détail). Les tweets ont donc été passés en minuscule, afin de permettre une meilleure correspondance avec les représentations pré-apprises de BERT et GPT. Les mentions d'utilisateurs, ainsi que les URL ont été remplacés par un tag, et ajouté au tokeniseur.

Concernant les hashtags, leurs fréquences d'apparition ont été vérifiées. Ainsi, les hashtags les plus fréquents ont été normalisés. Cela signifie que toutes les variations d'un même hashtag, « #buildthatwall » et « #buildthewall » par exemple, sont considérées comme équivalentes. Enfin, les abréviations les plus fréquentes ont été remplacées par leur forme complète [11].

5.6.2 Participation aux différentes tâches

La sous-tâche A de la tâche 5 de SemEval 2019, HatEval consistait à classifier des tweets en anglais comme étant haineux ou non. Après la création des différentes caractéristiques manuelles, les poids de certaines des représentations ont mis en avant que certaines de ces représentations pénalisaient la prédiction. Suivant la méthode de sélection de caractéristiques fondées sur le *meta-embedding* dynamique [72], ces caractéristiques ont été écartées afin de conserver uniquement celles qui avaient un intérêt pour la classification. Ainsi, les caractéristiques suivantes ont été utilisées :

— BERT pré-entraîné après un *fine-tuning* sur les éléments à prédire (haine, agressivité et cible), soit 3 représentations du texte de 768 caractéristiques [11].

— GPT pré-entraîné après un *fine-tuning* sur les éléments à prédire (haine, agressivité et cible), soit 3 représentations du texte de 768 caractéristiques [12].

— Décompte de mots présents dans un lexique de mots haineux extrait de HateBase [13].

— Décompte du nombre d'emojis regroupés par type.

Parmi la liste des caractéristiques créées manuellement et décrites plus haut dans ce chapitre, certaines n'ont pas été utilisées : celles-ci ne contribuaient pas, ou peu, aux performances du modèle. En effet, leurs pondérations étaient très faibles ou nulles, ainsi, elles ont été retirées du modèle pour cette tâche.

La sous-tâche B de la tâche 5 de SemEval 2019, HatEval consistait à prédire si chacun des tweets était agressif, et déterminer si la cible était générique ou s'il s'agissait d'un individu. Nous noterons que les tweets non-haineux été automatiquement étiquetés comme étant non-agressif et générique, ceci étant dû au processus d'annotation du jeu de données. Pour cette tâche, les mêmes caractéristiques que pour la sous-tâche A ont été choisies par le sélecteur de caractéristiques. Ainsi, cette fois encore, les mêmes caractéristiques ont été utilisées que pour la sous-tâche A.

Enfin, les sous-tâches A et B étaient également disponibles pour la langue espagnole. Les ressources étant plus rares pour cette langue, cette langue a été traitée en passant par l'anglais. La traduction automatique a permis de traduire chacun des tweets espagnol, et donc d'utiliser des ressources en anglais pour traiter cette tâche. Ainsi, le *meta-embedding* dynamique a une nouvelle fois été utilisé, en sélectionnant grâce à leur pondération les caractéristiques décrites précédemment, puisque les autres gênent la classification.

5.7 Conclusion du projet SAPhIRS

L'analyse d'un espace complexe comme les médias sociaux, combinée au phénomène de la radicalisation et de la haine, présente de nombreux défis. À l'issue de la troisième et dernière année, le projet SAPhIRS nous a amené à réaliser un démonstrateur capable

11. https://github.com/huggingface/pytorch-pretrained-BERT
12. https://github.com/huggingface/pytorch-openai-transformer-lm
13. https://hatebase.org/

de traiter de bout en bout des données de tweets, de la collecte à la visualisation, en passant par un ensemble de traitements algorithmiques et statistiques avancés.

De nombreux modèles d'apprentissage profond faiblement supervisés et des techniques d'analyse de graphes à l'état de l'art ont été développés et déployés en production, et ont été combinés en deux grandes chaînes de traitement : une chaîne de traitement pour détecter des discours de haine et des opinions, et une chaîne de traitement pour identifier des influenceurs et des communautés. L'ensemble des résultats est présenté dans des tableaux de bord dynamiques et interactifs.

Afin de valider l'opérationnalité du démonstrateur et vérifier l'adaptabilité du système à une thématique autre que la thématique initiale, une démonstration finale en direct a été préparée sur un mode défi sur des données réelles sous le pilotage de la DGA. Cette démonstration s'est portée sur les messages liés au boycott des produits Français sur la période de fin d'année 2020.

SAPhIRS est un succès pour l'ensemble des partenaires. Il nous a permis de développer et de consolider des compétences fortes en apprentissage profond faiblement supervisé pour l'analyse de textes courts, pour la détection d'opinion et en analyse de graphe. Nous avons valorisé ces travaux dans plusieurs articles scientifiques [3, 11, 37], et mettons à disposition les jeux de données utilisés à la communauté scientifique.

5.8 Synthèse

Cette section permet de faire un récapitulatif sur mes contributions au projet SAPhIRS ainsi que sur les publications scientifiques ayant eu lieu dans le cadre du projet. Parmi ces contributions et publications, j'ai notamment pu prendre part à la compétition SemEval 2019, et ainsi participer à la rédaction de l'article reprenant nos travaux.

Type	Contribution	Niveau de contribution
Module	Collecte de tweets et stockage	Moyen
Module	Détection de langue	Faible
Module	Détection de dialecte	Faible
Module	Translittération	Faible
Module	Détection de haine	Fort
Module	Détection d'objet d'opinion	Moyen
Module	Détection de polarité d'opinion	Fort
Pipeline	Mise en place de la chaîne de traitement de Détection de haine	Moyen
Visualisation	Visualisation des résultats	Faible
Compétition	Participation à la compétition SemEval	Moyen

TABLE 5.6 – Mes contributions dans le projet SAPhIRS (Fort : contributrice principale, Moyen : contribution équivalente entre les différents membres de l'équipe Recherche de Saagie, Faible : faible contribution

Compétition SemEval	(Benballa et al., 2019)[11]
Jeu de données LAD	(Baert et al., 2020)[3]
Détection de communautés et d'influenceurs	(Gadek, 2019)[37]

TABLE 5.7 – Liste des publications scientifiques dans le cadre du projet SAPhIRS

Chapitre 6

Conclusion

Au cours de cette thèse, nous nous sommes intéressés à l'analyse de sentiments sur des données issues de Twitter. En travaillant sur cette tâche, nous avons recensé différentes problématiques. Dans un premier temps, les modèles à l'état de l'art pour la classification de sentiments dans des tweets, fondés sur le Transformer [151], sont gourmands en données, mémoire, et surtout, en puissance de calcul. Ces ressources sont chères et difficilement accessibles. Afin de pouvoir effectuer de l'analyse de sentiments avec ces modèles à l'état de l'art, nous nous sommes alors penchés sur les lignes directrices à suivre pour réaliser notre tâche sans être freinés par le manque de ressources. Dans un second temps, en travaillant avec des données Twitter, nous avons été confrontés au langage spécifique contenu dans les messages de ce réseau social. En effet, les messages postés sur Twitter incluent un vocabulaire composé entre autres de hashtags, mentions d'utilisateur, et surtout d'emojis. De plus, comme la plupart des messages postés sur internet, les tweets sont souvent sarcastiques. La littérature a déjà démontré que prendre en compte les emojis augmentait les performances de l'analyse de sentiments. En revanche, dans le cadre de l'analyse de sentiments, la détection de sarcasme a principalement été utilisée pour inverser la polarité du sentiment dans les tweets sarcastiques. Nous nous sommes donc concentrés sur le traitement de cette information de sarcasme et d'emojis pour améliorer la classification de sentiments. Enfin, Twitter est également utilisé pour la propagation d'idéologies radicales et l'appel à la haine. Ainsi, dans le cadre du projet SAPhIRS, nous nous sommes intéressés au traitement de tweets sur des thématiques sensibles, comme la haine, la violence et la radicalisation.

Dans ce manuscrit, nous présentons des solutions aux différentes problématiques décrites ci-dessus. En utilisant le modèle BERT [33], référence parmi les modèles bidirectionnels fondés sur le Transformer, nous avons établi une liste de recommandations permettant de faire de la classification de tweets avec les méthodes à l'état de l'art, tout en ayant accès à des ressources limitées. Pour cela, nous nous sommes orienté vers le *fine-tuning* de modèles spécialisés lorsqu'ils sont disponibles, ou de modèles génériques. Pour de l'analyse de tweets, nous avons donc utilisé le modèle BERTweet [104], en le comparant au modèle BERT [33] générique. Nous avons dû contourner le problème d'oubli catastrophique auquel font face BERT-large et BERTweet, grâce au

gel de certaines couches du modèle et à l'utilisation d'un taux d'apprentissage faible. Ainsi, pour faire face au manque de ressources, nous recommandons l'utilisation de BERTweet, le modèle spécialisé, lorsque celui-ci est disponible, en restant attentif à sa potentielle perte de connaissances.

Puisque nos lignes directrices nous ont permis d'utiliser les modèles Transformer en ayant accès à des ressources limitées, nous avons souhaité aller plus loin et améliorer nos performances en classification du sentiment. Pour cela, nous nous sommes orienté vers l'utilisation d'un *embedding* d'emojis et d'une représentation du sarcasme. Ainsi, nous proposons de combiner le modèle BERTweet affiné sur de l'analyse de sentiments à une représentation d'emojis, Emojional, ainsi qu'au modèle BERT-base affiné sur de la détection de sarcasme. Cette fusion se décline de trois manières : tout d'abord, nous fusionnons BERTweet et Emojional, puis BERTweet et BERT, et enfin, nous combinons les trois représentation à la fois. Si l'ajout de la représentation d'emojis a permis d'améliorer la classification de sentiments, cette amélioration fut moins prononcée avec l'ajout des informations de sarcasme. Enfin, concernant la combinaison des trois éléments à la fois, les résultats furent encore moins bons, tout en restant supérieurs à l'utilisation de la *baseline* seule. Notre hypothèse est qu'il s'agit soit des informations de sarcasme qui pénalisent le modèle, soit les méthodes de combinaisons utilisées. En effet, nous n'avons employé que des méthodes de combinaison simples : la concaténation, la somme et la moyenne. Bien que la somme et la moyenne aient obtenu les meilleures performances, elles pourraient être inadaptées à la combinaison de caractéristiques aussi différentes qu'un *embedding* d'emojis et un espace latent représentant le sarcasme.

Enfin, dans le cadre du projet SAPhIRS, réalisé en partenariat entre Saagie, le LITIS et Airbus Defense and Space, nous nous sommes intéressés à la détection de radicalisation, de haine, de communautés et d'influenceurs sur des données en français, anglais et arabizi. Pour ce projet financé par la DGA, nous avons fourni deux chaînes de traitement : une première dédiée à la détection de communauté et d'influenceur et constituée des travaux d'Airbus et du LITIS, et une seconde dédiée à la détection de haine et de radicalisation regroupant les travaux de Saagie, et sur laquelle j'ai pu contribuer. Au cours de nos travaux sur ce projet, avec l'équipe de Recherche de Saagie, nous avons également pris part à la compétition SemEval 2019, pour la tâche de détection de haine en anglais et en espagnol. Nous y avons proposé un modèle fondé sur le *meta-embedding* dynamique, en fusionnant les représentations de BERT, GPT et d'autres caractéristiques créées manuellement. Les différentes représentations étaient combinées grâce à une approche semblable au *meta-embedding* dynamique, via l'utilisation d'une somme pondérée, ce qui a permis de filtrer les caractéristiques dont la pondération était la plus faible.

Si nos travaux nous ont permis de trouver des solutions à certains de nos problèmes, nous avons tout de même fait face à des difficultés et des limites. Dans un premier temps, en nous plaçant dans un contexte de ressources limitées, nous avons dû gérer le nombre important de paramètres pour BERT-large, et donc sa taille. En effet, le modèle est plus grand que BERT-base et BERTweet, nous avons donc dû adapter la taille de son batch et la longueur maximum de ses séquences afin que le modèle puisse

convenir aux 8 GB de RAM de notre GPU. Ainsi, plusieurs essais ont été réalisés afin de réduire ces deux hyper-paramètres sans pénaliser les performances du modèle. Dans un second temps, en étudiant l'impact de la prise en compte des emojis et de la détection de sarcasme sur l'analyse de sentiments, nous avons pu voir que le modèle BERT dédié au sarcasme, lorsqu'il est combiné à la représentation d'emojis et au modèle BERTweet affiné pour le sentiment, freinait la classification. En effet, bien que les résultats de cette combinaison soient meilleurs que notre *baseline*, ils restent inférieurs à ceux de BERTweet combinés seulement à la représentation d'emojis, qui a obtenu les meilleures performances de notre comparaison. En fonction de ces résultats, nous avons émis deux hypothèses à l'origine de cette limite : elle provient soit de la combinaison des différents éléments qui serait trop naïve et qui ne permettrait pas au modèle d'assimiler correctement les informations, soit d'un conflit entre les informations de la représentation d'emojis et du classifieur de sarcasme. Grâce à une étude plus approfondie, nous avons pu voir que la présence de sarcasme était ambiguë, même pour un humain, et que la majorité des tweets mal classifiés étaient détectés comme étant sarcastique. Nous avons ainsi obtenu une réponse partielle à notre problème : la présence de sarcasme gêne la classification. Cependant, l'étude des modes de combinaison reste à effectuer, car cela pourrait permettre au modèle d'intégrer et de traiter le sarcasme de manière différente et plus adaptée. Concernant le projet SAPhIRS, nous avons eu plusieurs difficultés à gérer. Tout d'abord, la constitution du jeu de données de radicalisation a subi beaucoup de retard. Ceci est dû à la fois à la thématique traitée, mais également à la tâche. En effet, à l'origine, l'objectif était la détection de changements d'opinion, et donc de pouvoir mettre en lumière le processus de radicalisation. Ce type de comportement étant difficile à détecter sur Twitter, la société ELDA chargée de collecter et annoter les données du projet a donc fait plusieurs tentatives infructueuses pour nous fournir des données d'entraînement conformes à notre cahier des charges. Si nous avons pu faire des regroupements concernant l'annotation en topiques des données, qui contenait beaucoup trop de catégories, l'annotation du changement d'opinion fut impossible à obtenir. Ces différents échanges avec la société ont à chaque fois repoussé l'obtention du jeu de données final et nous ont fait prendre du retard. La seconde difficulté concernant ce projet fut liée au traitement de l'Arabizi, alphabet de tchat arabe. En effet, très peu de ressources existent dans cette langue, ce qui a gêné l'apprentissage d'un modèle. Nous avons donc contourné ce problème en réalisant la translittération de l'Arabizi vers l'Arabe, puis de la traduction de l'Arabe vers l'Anglais. Cependant, avec cette méthode, nous perdons en précision, car la translittération n'est pas optimale, et les erreurs peuvent se cumuler en utilisant deux modèles différents.

Les travaux menés durant cette thèse peuvent être poursuivis dans différentes directions, dont trois nous paraissent primordiales. La première concerne l'utilisation des emojis pour l'analyse de sentiments dans des tweets. Dans ce manuscrit, la représentation des emojis a été limitée à l'utilisation de la démojisation et de l'*embedding* Emojional. Nous avons vu que l'intégration d'une représentation d'emojis améliore grandement la classification de sentiments. Nous pouvons donc émettre l'hypothèse qu'une représentation encore plus adaptée permettrait d'obtenir de meilleures performances en analyse de sentiments. Pour cela, nous envisageons l'utilisation d'un modèle de langue intégrant directement les emojis (et non leur versions démojisées comme

BERTweet), qui pourrait permettre de représenter les tweets intégralement. En effet, la démojisation déjà intégrée à BERTweet permet aux emojis d'être représentés de la même manière que le texte. Cependant, cette méthode nécessite une étape de normalisation au préalable qui augmente le temps de traitement du modèle. De plus, en transformant un emoji en texte, celui-ci va être découpé par le tokenizer avant d'être traité par le modèle. Or, nous pensons qu'il est plus pertinent de considérer les emojis comme étant des *tokens* à part entière, et donc de ne pas le découper. Ils pourront ainsi avoir une représentation unique, tout comme lors de l'utilisation de l'*embedding* Emojional. Cette approche d'intégration des emojis dans la représentation du texte présente différents avantages. Comme nous l'avons mentionné précédemment, cela permet de réduire le temps de traitement de chaque tweet, puisqu'il n'y a pas d'étape de normalisation des emojis. À cela s'ajoute le fait que les emojis possèdent une représentation qui leur est propre, et qui ne dépend plus des tokens utilisés pour le texte, sans avoir besoin de réaliser de combinaison. Cependant, une telle représentation des tweets, qui serait donc mixte, compliquerait la gestion de nouveaux emojis, contrairement à la démojisation. En effet, bien qu'il soit possible d'ajouter des *tokens* supplémentaires aux représentations, cela nécessite un entraînement afin d'apprendre les poids liés aux nouveaux *tokens*.

Notre seconde piste d'amélioration concerne la combinaison de modèles : en plus de l'utilisation d'un modèle unique pour classifier les tweets, nous souhaitons également tester la fusion de nos modèles avec des méthodes de combinaison différentes que celles décrites précédemment. En effet, les méthodes utilisées dans ce manuscrit sont simples, et nous pensons qu'elles pourraient être améliorées. Parmi les méthodes identifiées dans la littérature, nous nous intéressons particulièrement à l'utilisation de la somme pondérée. Cette approche est similaire à celle appliquée aux différentes caractéristiques utilisées lors de la compétition SemEval 2019. Cette méthode de combinaison pourrait être intéressante, car cela permettrait au modèle d'apprendre sa pondération et de mieux s'adapter aux tweets. Dans le cas des jeux de données utilisés pour nos expérimentations combinant l'analyse de sentiments, les emojis, et la détection de sarcasme, nous avons pu nous rendre compte que tous les tweets ne contenaient pas d'emojis. Dans ce genre de cas, avoir une pondération diminuerait l'impact de la représentation des emojis. Le modèle ne recevrait alors pas d'information inutile, tandis que les poids de la représentation du sarcasme et du sentiment seraient plus élevés. Avec une telle approche, il serait également possible d'ajouter différentes caractéristiques au modèle, par exemple une représentation spécifique aux hashtags, qui serait entraînée en amont en fonction de la thématique et de la période traitées. D'autres représentations du texte pourraient également être combinées, permettant ainsi de mieux capturer différentes caractéristiques des tweets. La gestion de ces différentes caractéristiques serait ensuite effectuée par la pondération du modèle : le contenu superflu serait ignoré, et les caractéristiques les plus importantes pour la tâche seront renforcées.

Enfin, notre troisième piste est directement liée au projet SAPhIRS et à la complexité d'annotation due aux thématiques de haine et de radicalisation. Nous avons pu constater que de tels sujets entraînent fréquemment un biais lors de l'étiquetage des données, ce qui détériore la qualité du classifieur. Parmi les méthodes permettant de détecter les erreurs dans l'annotation et de faire de la sélection de données, nous nous inté-

ressons aux méthodes de l'apprentissage confiant (*Confident Learning* [107]). Avec ce type d'approche, les exemples mal annotés sont tout d'abord mis en évidence. Puis, en fonction de la quantité de données disponibles ainsi que de la probabilité d'erreur, il est possible d'écarter ou encore de corriger les mauvaises annotations. Ainsi, l'apprentissage confiant est un moyen peu coûteux qui améliore la qualité des modèles de classification. Lors du traitement de données aussi sensibles que la radicalisation, il est difficile d'obtenir des données annotées. Corriger les erreurs d'étiquetage est donc une solution permettant de tirer parti des données mises à notre disposition, quelle que soit la qualité de leur annotation. Cependant, ce type d'approche n'est actuellement pas adapté aux données données multi-labels, et nous nous proposons d'étudier cette problématique de l'apprentissage confiant dans un cadre multi-labels.

Les différents points décrits ci-dessus seront explorés avec Saagie et dans le cadre du laboratoire commun créé entre Saagie et le LITIS.

Bibliographie

[1] Felipe Almeida and Geraldo Xexéo. Word embeddings : A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[2] Sai Saket Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. Deep learning models for multilingual hate speech detection. *arXiv preprint arXiv:2004.06465*, 2020.

[3] Gaétan Baert, Souhir Gahbiche, Guillaume Gadek, and Alexandre Pauchet. Arabizi language models for sentiment analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 592–603, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi : 10.18653/v1/2020.coling-main.51. URL https://aclanthology.org/2020.coling-main.51.

[4] Francesco Barbieri and Horacio Saggion. Automatic detection of irony and humour in twitter. In *ICCC*, pages 155–162, 2014.

[5] Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. Modelling sarcasm in Twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–58, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi : 10.3115/v1/W14-2609. URL https://aclanthology.org/W14-2609.

[6] Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. What does this emoji mean ? a vector space skip-gram model for twitter emojis. In *Calzolari N, Choukri K, Declerck T, et al, editors. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016) ; 2016 May 23-28 ; Portorož, Slovenia. Paris : European Language Resources Association (ELRA) ; 2016. p. 3967-72.* ELRA (European Language Resources Association), 2016.

[7] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd international conference on computational linguistics : posters*, pages 36–44. Association for Computational Linguistics, 2010.

[8] Elena Barry, Shoaib Jameel, and Haider Raza. Emojional : Emoji embeddings. In *UK Workshop on Computational Intelligence*, pages 312–324. Springer, 2021.

[9] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 task 5 : Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics", location = "Minneapolis, Minnesota, 2019.

[10] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. SemEval-2019 task 5 : Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/S19-2007. URL https://www.aclweb.org/anthology/S19-2007.

[11] Miriam Benballa, Sebastien Collet, and Romain Picot-Clemente. Saagie at semeval-2019 task 5 : From universal text embeddings and classical features to domain-specific text classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 469–475, 2019.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3 (Feb) :1137–1155, 2003.

[13] Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. Parsing-based sarcasm sentiment recognition in twitter data. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1373–1380. IEEE, 2015.

[14] Aashutosh Bhatt, Ankit Patel, Harsh Chheda, and Kiran Gawande. Amazon review classification and sentiment analysis. *International Journal of Computer Science and Information Technologies*, 6(6) :5107–5110, 2015.

[15] Alberto Bietti. Latent dirichlet allocation. Technical report, mai 2012, working paper, http://alberto. bietti. me/files/rapport-lda. pdf, 2012.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5 :135–146, 2017.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Ad-*

vances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[18] Gino Brunner, Y. Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. *arXiv : Computation and Language*, 2020.

[19] Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. Clues for detecting irony in user-generated contents : oh... !! it's" so easy" ;-. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56, 2009.

[20] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. Legal-bert : The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*, 2020.

[21] Yuxiao Chen, Jianbo Yuan, Quanzeng You, and Jiebo Luo. Twitter sentiment analysis via bi-sense emoji embedding and attention-based lstm. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 117–125, 2018.

[22] Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 2014.

[23] Varnith Chordia and Vijay Kumar BG. Large scale multimodal classification using an ensemble of transformer models and co-attention. *arXiv preprint arXiv:2011.11735*, 2020.

[24] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at ? an analysis of bert's attention. In *BlackboxNLP@ACL*, 2019.

[25] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra : Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

[26] Mathieu Cliche. Bb_twtr at semeval-2017 task 4 : Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*, 2017.

[27] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, F. Viégas, and M. Wattenberg. Visualizing and measuring the geometry of bert. In *NeurIPS*, 2019.

[28] Yimian Dai, Fabian Gieseke, Stefan Oehmcke, Yiquan Wu, and Kobus Barnard. Attentional feature fusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3560–3569, 2021.

[29] Zihang Dai, Zhilin Yang, Yiming Yang, William W. Cohen, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. Transformer-xl : Language modeling with longer-term dependency. 2018.

[30] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery : Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528, 2003.

[31] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116, 2010.

[32] Pieter Delobelle and Bettina Berendt. Time to take emoji seriously : They vastly improve casual conversational models. *arXiv preprint arXiv:1910.13793*, 2019.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[34] Javid Ebrahimi, Dejing Dou, and Daniel Lowd. A joint sentiment-target-stance model for stance classification in tweets. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, pages 2656–2665, 2016.

[35] Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. emoji2vec : Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 48–54, Austin, TX, USA, November 2016. Association for Computational Linguistics. doi : 10.18653/v1/W16-6208. URL https://aclanthology.org/W16-6208.

[36] E. Fersini, P. Rosso, and M. Anzovino. Overview of the task on automatic misogyny identification at ibereval 2018. In *IberEval@SEPLN*, 2018.

[37] Guillaume Gadek. From community detection to topical, interactive group detection in online social networks. In *IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume*, WI '19 Companion, page 176–183, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369886. doi : 10.1145/3358695.3360894. URL https://doi.org/10.1145/3358695.3360894.

[38] Guillaume Gadek, Josefin Betsholtz, Alexandre Pauchet, Stéphan Brunessaux, Nicolas Malandain, and Laurent Vercouter. Extracting contextonyms from twitter for stance detection. In *International Conference on Agents and Artificial*

Intelligence, volume 2, pages 132–141. SCITEPRESS, 2017.

[39] Guillaume Gadek, Alexandre Pauchet, Nicolas Malandain, Khaled Khelif, Laurent Vercouter, and Stéphan Brunessaux. Measures for topical cohesion of user communities on twitter. In *Proceedings of the International Conference on Web Intelligence*, pages 211–218, 2017.

[40] Guillaume Gadek, Alexandre Pauchet, Nicolas Malandain, Khaled Khelif, Laurent Vercouter, and Stéphan Brunessaux. Topical cohesion of communities on twitter. *Procedia computer science*, 112 :584–593, 2017.

[41] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2) : 23–38, 1994.

[42] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-dependent sentiment classification with bert. *IEEE Access*, 7 :154290–154299, 2019.

[43] Manoochehr Ghiassi, David Zimbra, and Sean Lee. Targeted twitter sentiment analysis for brands using supervised feature engineering and the dynamic architecture for artificial neural networks. *Journal of Management Information Systems*, 33(4) :1034–1058, 2016.

[44] Hatem Ghorbel and David Jacot. Further experiments in sentiment analysis of french movie reviews. In *Advances in Intelligent Web Mastering–3*, pages 19–28. Springer, 2011.

[45] Aniruddha Ghosh and Tony Veale. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 161–169, 2016.

[46] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[47] Alec Go, Lei Huang, and Richa Bhayani. Twitter sentiment analysis. *Entropy*, 17 :252, 2009.

[48] Y. Goldberg. Assessing bert's syntactic abilities. *ArXiv*, abs/1901.05287, 2019.

[49] Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in Twitter : A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 581–586, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-2102.

[50] Gaël Guibon, Magalie Ochs, and Patrice Bellot. From emojis to sentiment analysis. In *WACAI 2016*, 2016.

[51] Tanjim Ul Haque, Nudrat Nawal Saber, and Faisal Muhammad Shah. Sentiment

analysis on large scale amazon product reviews. In *2018 IEEE international conference on innovative research and development (ICIRD)*, pages 1–6. IEEE, 2018.

[52] Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. CASCADE : Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1837–1848, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1156.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[54] I Hemalatha, GP Saradhi Varma, and A Govardhan. Sentiment analysis tool using machine learning algorithms. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 2(2) :105–109, 2013.

[55] Irazú Hernández-Farías, José-Miguel Benedí, and Paolo Rosso. Applying basic features from sentiment analysis for automatic irony detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 337–344. Springer, 2015.

[56] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. Aspect-based sentiment analysis using bert. In *NEAL Proceedings of the 22nd Nordic Conference on Computional Linguistics (NoDaLiDa), September 30-October 2, Turku, Finland*, number 167, pages 187–196. Linköping University Electronic Press, 2019.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[58] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. exBERT : A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, pages 187–196, Online, July 2020. Association for Computational Linguistics. doi : 10.18653/v1/2020.acl-demos.22. URL https://aclanthology.org/2020.acl-demos.22.

[59] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi : 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031.

[60] Clayton Hutto and Eric Gilbert. Vader : A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[61] Tunazzina Islam and Dan Goldwasser. Does yoga make you happy ? analyzing twitter user happiness using textual and temporal information. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4241–4249. IEEE, 2020.

[62] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *NAACL-HLT*, 2019.

[63] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language ? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356.

[64] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart : Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*, 2019.

[65] Tanimu Ahmed Jibril and Mardziah Hayati Abdullah. Relevance of emoticons in computer-mediated communication contexts : An overview. *Asian Social Science*, 9(4) :201, 2013.

[66] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[67] Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2 : Short Papers)*, pages 757–762, 2015.

[68] Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection ? *arXiv preprint arXiv:1610.00883*, 2016.

[69] Jacqueline Kazmaier and Jan H van Vuuren. The power of ensemble learning in sentiment analysis. *Expert Systems With Applications*, 187 :115819, 2022.

[70] Vishal Kharde, Prof Sonawane, et al. Sentiment analysis of twitter data : A survey of techniques. *arXiv preprint arXiv:1601.06971*, 2016.

[71] Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. Your sentiment precedes you : Using an author's historical tweets to predict sarcasm. In *Proceedings of the 6th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 25–30, 2015.

[72] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the*

2018 Conference on Empirical Methods in Natural Language Processing, pages 1466–1477, 2018.

[73] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi : 10.3115/v1/D14-1181. URL https://aclanthology.org/D14-1181.

[74] Dilek Küçük and Fazli Can. Stance detection : A survey. *ACM Computing Surveys (CSUR)*, 53(1) :1–37, 2020.

[75] Alwine Lambert, Gabriel Bellard, Guillaume Lorre, and Karim Kouki. Analyse de sentiment twitter. 2016.

[76] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert : A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[77] Jörg Landthaler, Bernhard Waltl, Dominik Huth, Daniel Braun, Christoph Stocker, Thomas Geiger, and Florian Matthes. Extending thesauri using word embeddings and the intersection method. *ASAIL@ ICAIL*, 8(1) :112–119, 2017.

[78] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. Flaubert : Unsupervised language model pre-training for french. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France, May 2020. European Language Resources Association. URL https://www.aclweb.org/anthology/2020.lrec-1.302.

[79] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) : 2278–2324, 1998.

[80] Jaejun Lee, Raphael Tang, and Jimmy J. Lin. What would elsa do ? freezing layers during transformer fine-tuning. *ArXiv*, abs/1911.03090, 2019.

[81] HR Lekshmiammal and Anand Kumar Madasamy. Nitk _ nlp at checkthat ! 2021 : Ensemble transformer model for fake news classification. In *Conference and Labs Ofthe Evaluation Forum (CLEF 2021)*, 2021.

[82] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. Exploiting bert for end-to-end aspect-based sentiment analysis. *arXiv preprint arXiv:1910.00883*, 2019.

[83] Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37, Atlanta, Georgia, June 2013. Association for Computa-

tional Linguistics. URL https://aclanthology.org/W13-1605.

[84] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE, 2015.

[85] Yongjie Lin, Y. Tan, and R. Frank. Open sesame : Getting inside bert's linguistic knowledge. In *BlackboxNLP@ACL*, 2019.

[86] Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2 :627–666, 2010.

[87] Chuchu Liu, Fan Fang, Xu Lin, Tie Cai, Xu Tan, Jianguo Liu, and Xin Lu. Improving sentiment analysis accuracy with emoji embedding. *Journal of Safety Science and Resilience*, 2(4) :246–252, 2021.

[88] Shenghua Liu, Fuxin Li, Fangtao Li, Xueqi Cheng, and Huawei Shen. Adaptive co-training svm for sentiment classification on tweets. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2079–2088, 2013.

[89] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta : A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[90] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems*, 29, 2016.

[91] Zhunchen Luo, Miles Osborne, and Ting Wang. An effective approach to tweets opinion retrieval. *World Wide Web*, 18(3) :545–566, 2015.

[92] Navonil Majumder, Soujanya Poria, Haiyun Peng, Niyati Chhaya, Erik Cambria, and Alexander Gelbukh. Sentiment and sarcasm classification with multitask learning. *IEEE Intelligent Systems*, 34(3) :38–43, 2019.

[93] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de La Clergerie, Djamé Seddah, and Benoît Sagot. Camembert : a tasty french language model. *arXiv preprint arXiv:1911.03894*, 2019.

[94] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[95] Reza Maulana, Panny Agustia Rahayuningsih, Windi Irmayani, Dedi Saputra, and Wanty Eka Jayanti. Improved accuracy of sentiment analysis movie review using support vector machine based information gain. In *Journal of Physics :*

Conference Series, volume 1641, page 012060. IOP Publishing, 2020.

[96] Diana G Maynard and Mark A Greenwood. Who cares about sarcastic tweets ? investigating the impact of sarcasm on sentiment analysis. In *Lrec 2014 proceedings*. ELRA, 2014.

[97] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks : The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi : https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[98] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[99] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[100] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. SemEval-2016 task 6 : Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, June 2016. Association for Computational Linguistics. doi : 10.18653/v1/S16-1003. URL https://aclanthology.org/S16-1003.

[101] Anirban Mukherjee, Sabyasachi Mukhopadhyay, Prasanta K Panigrahi, and Saptarsi Goswami. Utilization of oversampling for multiclass sentiment analysis on amazon review dataset. In *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, pages 1–6. IEEE, 2019.

[102] Martin Müller, Marcel Salathé, and Per E Kummervold. Covid-twitter-bert : A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*, 2020.

[103] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. SemEval-2016 task 4 : Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June 2016. Association for Computational Linguistics.

[104] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet : A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, pages 9–14, Online, October 2020. Association for Computational Linguistics. doi : 10.18653/v1/2020.emnlp-demos.2. URL https://aclanthology.org/2020.emnlp-demos.2.

[105] Giang Nguyen, Shuan Chen, T. Jun, and Daeyoung Kim. Explaining how deep

neural networks forget by deep visualization. In *ICPR Workshops*, 2020.

[106] Ru Ni and Huan Cao. Sentiment analysis based on glove and lstm-gru. In *2020 39th Chinese Control Conference (CCC)*, pages 7492–7497. IEEE, 2020.

[107] Curtis Northcutt, Lu Jiang, and Isaac Chuang. Confident learning : Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70 : 1373–1411, 2021.

[108] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq : A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

[109] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[110] Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, Viviana Patti, et al. 14-exlab@ unito for ami at ibereval2018 : Exploiting lexical knowledge for detecting misogyny in english and spanish tweets. In *3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages, IberEval 2018*, volume 2150, pages 234–241. CEUR-WS, 2018.

[111] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up ?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[112] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2) :1–135, 2008.

[113] Ji Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*, 2017.

[114] Krishna Parmar, Nivid Limbasiya, and Maulik Dhamecha. Feature based composite approach for sarcasm detection using mapreduce. In *2018 second international conference on computing methodologies and communication (ICCMC)*, pages 587–591. IEEE, 2018.

[115] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[116] Keval Pipalia, Rahul Bhadja, and Madhu Shukla. Comparative analysis of different transformer based architectures used in sentiment analysis. In *2020 9th International Conference System Modeling and Advancement in Research Trends*

(SMART), pages 411–415. IEEE, 2020.

[117] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*, 2016.

[118] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[119] Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[120] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[121] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2020.

[122] Callen Rain. Sentiment analysis in amazon reviews using probabilistic machine learning. *Swarthmore College*, 2013.

[123] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. Sarcasm detection on twitter : A behavioral modeling approach. In *Proceedings of the eighth ACM international conference on web search and data mining*, pages 97–106, 2015.

[124] Antonio Reyes, Paolo Rosso, and Davide Buscaldi. From humor recognition to irony detection : The figurative language of social media. *Data & Knowledge Engineering*, 74 :1–12, 2012.

[125] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 704–714, 2013.

[126] Aiala Rosá, Luis Chiruzzo, Mathias Etcheverry, and Santiago Castro. Retuyt en tass 2017 : Análisis de sentimientos de tweets en espanol utilizando svm y cnn. *Proceedings of TASS*, 2017.

[127] Sara Rosenthal, Noura Farra, and Preslav Nakov. SemEval-2017 task 4 : Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi : 10.18653/v1/

S17-2088. URL https://aclanthology.org/S17-2088.

[128] Sebastian Ruder. *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway, 2019.

[129] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1 : Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[130] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[131] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[132] Edoardo Savini and Cornelia Caragea. Intermediate-task transfer learning with bert for sarcasm detection. *Mathematics*, 10(5) :844, 2022.

[133] Kim Schouten and Flavius Frasincar. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3) :813–830, 2015.

[134] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[135] Ovidiu Serban, Alexandre Pauchet, Alexandrina Rogozan, Jean-Pierre Pécuchet, and INSA LITIS. Semantic propagation on contextonyms using sentiwordnet. In *WACAI 2012, Workshop Affect, Compagnon Artificiel, Interaction*, page 86, 2012.

[136] Ovidiu Serban, Alexandre Pauchet, Alexandrina Rogozan, and Jean-Pierre Pecuchet. Modelling context to solve conflicts in sentiwordnet. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 393–398. IEEE, 2013.

[137] Sofia Serrano and Noah A. Smith. Is attention interpretable ? *ArXiv*, abs/1906.03731, 2019.

[138] Mohammed Shiha and Serkan Ayvaz. The effects of emoji in sentiment analysis. *Int. J. Comput. Electr. Eng.(IJCEE.)*, 9(1) :360–369, 2017.

[139] Abhishek Singh, Eduardo Blanco, and Wei Jin. Incorporating emoji descriptions improves tweet classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human*

Language Technologies, Volume 1 (Long and Short Papers), pages 2096–2101, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/N19-1214. URL https://aclanthology.org/N19-1214.

[140] Abhishek Singh, Eduardo Blanco, and Wei Jin. Incorporating emoji descriptions improves tweet classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2096–2101, 2019.

[141] Himani Srivastava, Vaibhav Varshney, Surabhi Kumari, and Saurabh Srivastava. A novel hierarchical bert architecture for sarcasm detection. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 93–97, 2020.

[142] Yuanhang Su and C-C Jay Kuo. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, 356 :151–161, 2019.

[143] Jayashree Subramanian, Varun Sridharan, Kai Shu, and Huan Liu. Exploiting emojis for sarcasm detection. In *SBP-BRiMS*, 2019.

[144] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification ? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[145] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[146] Tiancheng Tang, Xinhuai Tang, and Tianyi Yuan. Fine-tuning bert for multi-label sentiment analysis in unbalanced code-switching text. *IEEE Access*, 8 : 193248–193256, 2020.

[147] Joseph Tepperman, David Traum, and Shrikanth Narayanan. " yeah right" : Sarcasm recognition for spoken dialogue systems. In *Ninth international conference on spoken language processing*, 2006.

[148] Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*, 36(6) :823–848, 2010.

[149] Oren Tsur, Dmitry Davidov, and Ari Rappoport. Icwsm—a great catchy name : Semi-supervised recognition of sarcastic sentences in online product reviews. In *fourth international AAAI conference on weblogs and social media*, 2010.

[150] Cynthia Van Hee, Els Lefever, and Véronique Hoste. SemEval-2018 task 3 : Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana, June

2018. Association for Computational Linguistics. doi : 10.18653/v1/S18-1005. URL https://aclanthology.org/S18-1005.

[151] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[152] Tony Veale and Yanfen Hao. Detecting ironic intent in creative comparisons. In *ECAI 2010*, pages 765–770. IOS Press, 2010.

[153] S Vijayarani, Ms J Ilamathi, Ms Nithya, et al. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1) :7–16, 2015.

[154] Vishal.A.Kharde and Prof. Sheetal.Sonawane. Sentiment analysis of twitter data : A survey of techniques. 2016.

[155] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention : Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

[156] Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 512–516, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi : 10.3115/v1/P14-2084. URL https://aclanthology.org/P14-2084.

[157] Byron C. Wallace, Do Kook Choe, and Eugene Charniak. Sparse, contextually informed models for irony detection : Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 1035–1044, Beijing, China, July 2015. Association for Computational Linguistics. doi : 10.3115/v1/P15-1100. URL https://aclanthology.org/P15-1100.

[158] Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2 : Short papers)*, pages 225–230, 2016.

[159] Sida I Wang and Christopher D Manning. Baselines and bigrams : Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 90–94, 2012.

[160] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. Automated concatenation of embeddings for structured prediction. *arXiv preprint arXiv:2010.05006*, 2020.

[161] Hans Weytjens and Jochen De Weerdt. Process outcome prediction : Cnn vs. lstm (with attention). In *International Conference on Business Process Management*, pages 321–333. Springer, 2020.

[162] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP*, 2019.

[163] Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system : Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

[164] Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*, 2016.

[165] Yi Yang, Mark Christopher Siy Uy, and Allen Huang. Finbert : A pretrained language model for financial communications. *arXiv preprint arXiv:2006.08097*, 2020.

[166] Kuat Yessenov and Saša Misailovic. Sentiment analysis of movie review comments. *Methodology*, 17 :1–7, 2009.

[167] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[168] Yanping Yin and Zhong Jin. Document sentiment classification based on the word embedding. In *2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering*. Atlantis Press, 2015.

[169] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/N19-1144. URL https://aclanthology.org/N19-1144.

[170] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra,

and Ritesh Kumar. SemEval-2019 task 6 : Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/S19-2010. URL https://aclanthology.org/S19-2010.

[171] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 8(4) :e1253, 2018.

[172] Meishan Zhang, Yue Zhang, and Guohong Fu. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : technical papers*, pages 2449–2460, 2016.

[173] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[174] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[175] Zichen Zhang, Yuhang Wu, and Hao Wu. Yun_de at hasoc2020 subtask a : Multimodel ensemble learning for identifying hate speech and offensive language. In *FIRE (Working Notes)*, pages 217–223, 2020.

[176] Yukun Zhu, Ryan Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies : Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.