De l'analyse statistique de données textuelles aux réseaux de neurones artificiels. Vers des motifs linguistiques profonds

Laurent Vanni

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

De l'analyse statistique de données textuelles aux réseaux de neurones artificiels.
Vers des motifs linguistiques profonds

Laurent Vanni

UMR 7320 : Bases, Corpus, Langage - Nice

UMR 7271 : Informatique, Signaux et Systèmes - Sophia-Antipolis

Présentée en vue de l'obtention du grade de docteur en Informatique d' Université Côte d'Azur

Dirigée par :
Damon Mayaffre - CR CNRS
Frédéric Précioso - PU Université Côte d'Azur

Devant le jury, composé de :
Président du jury :
Fabrice Huet - PU Université Côte d'Azur

Rapportrices :
Valérie Beaudouin - PU Télécom Paris
Florence Sèdes - PU Université Toulouse III

Examinateur :
Dominique Longrée - PU Université de Liège

Soutenue le : 10 décembre 2021

« Using the arsenal of black boxes available, the machine learns. Using the large subset of selected techniques, the researcher learns. »

L. Lebart

Résumé

Au croisement de la statistique textuelle, de l'analyse automatique des langues et du *deep learning*, cette thèse propose une méthodologie nouvelle pour l'analyse des texte. L'hypothèse de départ repose sur l'architecture des réseaux de neurones et leur capacité à extraire automatiquement l'information contenue dans un texte. La précision des modèles convolutionnels pour la classification de textes souligne l'importance des marqueurs linguistiques identifiés. Pour rendre accessible aux linguistes cette information d'un genre nouveau nous développons un protocole d'analyse qui combine l'approche informatique avec l'approche linguistique. Plus particulièrement :

1) La deconvolution Zeiler et Fergus 2014, procédé algorithmique utilisé dans l'analyse d'images, est appliquée aux textes pour pouvoir décoder la représentation des mots engendrés par le réseau de neurones. Plus généralement nous nous intéressons à l'ensemble des couches d'un modèle profond pour comprendre et transmettre aux linguistes la chaine de traitements qui existe entre les données brutes (le texte) en entrée du réseau et la prédiction en sortie du réseau (classification). Une nouvelle mesure est proposée pour rendre compte du poids des mots dans un modèle profond : le Text Deconvolution Saliency (**TDS**)

2) Les *motifs* linguistiques fortement théorisés par Mellet et D. Longrée 2009 accompagnent notre raisonnement sur les réseaux de neurones. Leur nature nous pousse à concevoir des architectures profondes capables de considérer le texte comme un objet complexe et multidimensionnel. Les motifs nous conduisent à entrevoir des *passages-clés* à la sortie d'une déconvolution et des *motifs profonds* lorsque le modèle utilise plusieurs représentations des mots (forme graphique, catégorie grammaticale, lemme).

Ce lien que nous tentons de créer entre observations empiriques (**TDS**) et théories linguistiques ouvre des voies nouvelles d'interprétation des textes. Les cas d'utilisation de notre méthode sont nombreux et font déjà l'objet de plusieurs études que nous verrons tout au long de cette thèse. L'exploration des réseaux de neurones profonds pour la linguistique de corpus n'en est encore qu'à ses débuts, mais les premiers résultats que nous présentons ici sont dès à présent encourageants.

Mots clés : texte, motif, deep learning, statistique, linguistique, analyse de données textuelles

Abstract

This thesis presents a new methodology for text analysis which is situated at the intersection of textual statistics, automatic language analysis and deep learning. It draws on the architecture of neural networks and its potential to extract information from texts. The accuracy of convolutional models for text classification depends on the quality of the linguistic markers which are identified. To make these markers accessible to linguists, we implement an algorithm that combines the following computational and linguistic approaches :

1) Deconvolution (Zeiler et Fergus 2014) is an algorithmic process used in image analysis which we apply here to written texts in order to interpret the representations of words which are learned by the neural network. In particular, we analyse each layer of a deep model to understand the mechanisms linking the raw data which forms the input to the network (the text) with the predictions which constitute its output (classification). The aim is to express these connections in ways which are useful to linguists. We propose a new measure to express the relative weight of words in a deep model : Text Deconvolution Saliency (**TDS**).

2) Our approach also comprises a linguistic theory of textual motifs or patterns developed by Mellet et D. Longrée 2009 which informs the design of deep architectures which allow the analysis of texts as complex, multidimensional linguistic objects. The motif theory permits the detection of key passages in the output of deconvolution, and even the identification of deep motifs in cases where the model propose suggests several representations of words (e.g., full-form, part-of-speech, and lemma).

The new methodology developed in our research connects empirical observations (**TDS**) with linguistic analysis in ways which open up new possibilities for the interpretation of texts. A number of studies of the application of this protocol in different contexts have served to test the methodology and will be used to illustrate its effectiveness throughout this doctoral thesis. Although deep neural network interpretation for linguistics analysis is a very new field, the initial results presented here are encouraging, and the thesis concludes with suggestions for further research in this domain.

Keywords : texte, motif, deep learning, statistique, linguistique, analyse de données textuelles

Remerciements

Avant tout, je tiens à remercier les membres du jury pour leur relecture et l'appréciation interdisciplinaire qu'ils ont faites de mon travail. Plus particulièrement à Fabrice Huet pour avoir accepté de présider ce jury. À Valérie Beaudouin pour son expertise en Analyse de Données Textuelle. À Florence Sèdes son évaluation en tant que spécialiste en informatique. Et à Dominique Longrée pour son soutien scientifique et amical important depuis le début de ces travaux.

Mon entière gratitude va ensuite à mes directeurs de thèse. Merci à Frédéric Précioso pour son engagement, sa confiance et ses conseils tout au long de cette thèse. Merci à Damon Mayaffre sans qui cette thèse n'existerait pas. Sa rencontre a été déterminante pour moi. Ses travaux, son savoir et plus généralement le bon sens et le discernement dont il fait preuve forcent une admiration qui dépasse pour moi le seul cadre de cette thèse. Merci pour sa présence, son soutien, j'espère continuer à apprendre et à me former à ses côtés pendant de nombreuses années.

Je tiens aussi à remercier Étienne Brunet qui, comme Damon, est pour moi un exemple d'excellence scientifique et un modèle de longévité intellectuelle. Merci de m'avoir encouragé à m'inscrire en thèse et laissé aussi utiliser et abuser de son logiciel et du nom qu'il lui a donné, Hyperbase.

Cette thèse a été aussi pour moi l'occasion de rencontres heureuses qui ont abouti à des productions scientifiques de qualité. Je remercie le laboratoire I3S (UMR7271) pour son accueil et plus particulièrement Mélanie Ducoffe qui m'a offert une formation accélérée en deep learning et qui m'a accompagné jusqu'en Australie pour présenter le fruit de nos travaux communs. Je remercie aussi Marco Corneli de la MSI pour son expertise mathématique et nos échanges scientifiques et amicaux enrichissants.

Merci à l'équipe *Logométrie* de m'avoir accueilli et formé depuis 2013 en linguistique de corpus. Merci à Céline Poudat pour son animation et les nombreux séminaires et formations que j'ai pu suivre ou proposer. Merci à Magali Guaresi d'avoir été la première doctorante à bien vouloir tester les méthodes et outils que je propose. Merci aussi à Véronique Magri, Simona Ruggia, Camille Bouzereau, Federica Beghini qui m'ont aidé à perfectionner les applications et qui ont fait le pari durant cette thèse d'intégrer le deep learning dans leurs recherches. Plus généralement, un immense merci aux utilisateurs d'Hyperbase Web qui me font confiance et qui font vivre la plateforme depuis sa mise en ligne en 2015.

Merci également aux collègues et camarades de mon laboratoire de rattachement - Bases, Corpus, Langage (UMR7320) - pour leur soutien et leur sympathie. Merci à la direction du laboratoire, Fanny Meunier et Richard Faure pour m'avoir permis d'entreprendre un parcours doctoral tout en poursuivant mes fonctions d'ingénieur de recherche au CNRS. Merci aussi à Caroline Daire, Delphine Chetiveaux et Odile Deangeli pour leur accompagnement administratif et leur bonne humeur. Merci à Pierre-Aurélien Georges avec qui je partage mon bureau et qui m'offre une aide technique précieuse. Merci à Mustapha,

Alexander, Marie, Jonathan et tous ceux avec qui j'ai partagé des moments de convivialité autour de la machine à café.

Enfin, je veux remercier mes proches et amis pour leur soutien permanent. Mes parents pour m'avoir toujours encouragé. Sandrine, Milo et Julie pour l'équilibre et le recul qu'ils m'apportent au quotidien.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

« On prête au Père Busa[1], au sortir de la 2e guerre mondiale, la première entreprise majeure croisant textes et informatique. Depuis lors, implémentations numériques, propositions logicielles et méthodes algorithmiques se sont multipliées pour faire parler les textes, les décrire ou les modéliser » (Vanni et Precioso 2021).

Dans ce contexte, cette thèse introduit un protocole d'analyse de données textuelles basé sur l'apprentissage des réseaux neurones profonds. La méthode présentée offre une plus-value interprétative aux linguistes et complètent l'approche statistique traditionnelle par un modèle de *deep learning* qui se veut dédié à la description des textes. Les résultats présentés prennent la forme de contributions innédites partagés entre deux disciplines, l'informatique et les sciences humaines et sociales, plus particulièrement les réseaux de neurones profonds et la linguistique de corpus (figure B.1).

1.1. Contributions

La trame de fond de cette thèse repose avant tout sur des développements informatiques qui mêlent différents aspects du *deep learning* appliqués aux textes. L'hypothèse de départ est que de l'interprétabilité de ces méthodes dépend les nouveaux usages en SHS.

1.1.1. Informatique

Les modèles étudiés tout au long de ces travaux se focalisent sur des architectures à convolution (voir les prérequis, section 2.3.2). Ce choix est guidé notamment par les performances de ce type de modèles dans le domaine de la classification de textes qui est notre point de convergence méthodologique entre *deep learning* et méthodes statistiques plus classiques.

*Text Deconvolution Saliency (**TDS**)*

La contribution principale de cette thèse concerne la notion de *déconvolution* (section 3.2). L'application de cette méthode à l'analyse de textes (initialement développée pour le traitement des images), offre des moyens nouveaux pour interpréter linguistiquement le comportement d'un réseau de neurones. La représentation des textes obtenu par déconvo-

1. Roberto Busa, prêtre jésuite italien, fut un des pionniers dans l'utilisation de l'ordinateur pour l'analyse linguistique et littéraire. Il a notamment été le premier à créer un thesaurus en collaboration avec IBM (sur l'œuvre de Thomas d'Aquin), un travail entamé en 1949 et qui s'est terminé trente ans plus tard.

lution offre de nombreuses perspectives en analyse de données textuelles et se caractérise par la proposition d'une mesure que nous appelons, le *Text Deconvolution Saliency* (**TDS**) (Vanni, Ducoffe et al. 2018 ; Vanni, Mayaffre et D. Longrée 2018 ; Vanni et Precioso 2021).

Text Class Activation Map (T-CAM)

La deuxième contribution présentée améliore considérablement les résultats du **TDS**. Inspiré aussi du traitement des images, cet algorithme que nous appelons *Text Class Activation Map* (chapitre 4) corrige les défauts du **TDS** avec une version pondérée de la mesure, le *weighted Text Deconvolution Saliency* (**wTDS**), qui propose un niveau d'interprétation plus fin. (Vanni, Corneli, D. Longrée et al. 2020a ; Vanni, Corneli, F. Precioso et al. 2022).

Déconvolution multi-channels

La détection automatique de marqueurs linguistiques complexes (les *motifs* profonds), sorte de nouveaux observables du texte, introduit des notions de linguistique multidimensionnelles. En d'autres termes, les mots ne sont pas seulement considérés comme des formes graphiques (une suite de caractères qui représente le mot dans le texte), ils sont aussi associés à différentes représentations comme une catégorie grammaticale ou un lemme (forme canonique du mot). Cette contrainte est à l'origine d'une autre extension du **TDS** (chapitre 6) : nous passons alors d'une convolution simple (*mono-channel*, un seul type de représentation des données) à une convolution multiniveaux (ou *multi-channels*). Cette architecture particulière associée au **wTDS** autorise la détection de marqueurs multicritères dans le texte et ouvre sur la détection automatique de *motifs* profonds (Vanni, Corneli, D. Longrée et al. 2020a ; Vanni et Precioso 2021).

Applications

Toutes les méthodes évoquées précédemment se matérialisent aussi par des applications informatiques concrètes (chapitre 8). Le moteur principal *Hyperdeep* (section 8.2), à l'origine de ces applications, implémente les algorithmes proposés dans cette thèse et permet le développement d'outils qui diffusent la méthode au sein de la communauté des sciences humaines. Ici le défit est avant tout technique puisque le *deep learning*, gourmand en termes de ressources et complexe en termes de concepts, est proposé sous forme de services web associés à une ergonomie intelligible pour un utilisateur non informaticien attiré avant tout par la portée linguistique de ces outils (Vanni, Corneli, D. Longrée et al. 2020a ; Ruggia et Vanni 2021).

1.1.2. Linguistique

Le travail réalisé côté informatique apporte des contributions quasi symétriques en sciences humaines. La description des textes par le *deep learning* élargit le champ de recherche en linguistique de corpus et alimente certaines questions laissées en suspens par la statistique classique.

Texte : objet complexe

Selon Noam Chomsky il existe une grammaire universelle constituée d'un ensemble de règles que les humains ont la capacité (innée) d'acquérir. Si la linguistique chomskyienne

aide à comprendre la structure locale des textes (les phrases, les paragraphes ...), elle ne renseigne pas sur une structure plus globale, le texte dans son ensemble comme un objet complexe, *cohésif* et *cohérent* (Halliday et Hasan 1976 ; Adam 2011). Qu'est-ce qui fait un texte ? Nous nous intéressons ici à la notion de *passage* et par extension de *passages-clés* (chapitre 5). Le deep learning semble sensible à des unités linguistiques souples. Du mot seul à la cooccurrence de mots ou d'expressions, les modèles utilisent toutes les informations disponibles dans un corpus d'entrainement pour apprendre ce qui fait un texte. Une fois restituée par le **wTDS**, cette analyse (ou plutôt description) euristique des textes par la machine interroge sur la question du texte comme objet complexe (Vanni, Corneli, D. Longrée et al. 2020b ; Vanni, Mayaffre et D. Longrée 2018 ; Mayaffre et Vanni 2021a).

Grandeurs textuelles : motifs profonds

Les motifs pour (Mellet et D. Longrée 2009) sont des unités linguistiques multidimensionnelles complexes qui caractérisent les textes (plus généralement une métadonnée comme un auteur, une année ou un genre). Or le *deep learning* est capable de considérer les textes comme des ensembles d'unités complexes avec plusieurs niveaux de représentations [2]. Dans le chapitre 6 les motifs dits *profonds* détectés par la machine questionnent l'approche uniquement théorique jusqu'ici proposée (Mayaffre et Vanni 2021b).

Intertexte : corpus réflexif

L'intertextualité est un autre sujet que nous tentons d'objectiver dans ces travaux. En proposant un parcours méthodologique nouveau (chapitre 7) cette thèse montre ici que le deep learning est un moyen observer les phénomènes d'emprunt, d'imitation voire de plagiat dans les textes. En utilisant les capacités prédictives des réseaux convolutionnels, nous verrons une méthode qui permet d'extraire et de décrire l'intertexte dans un corpus (Mayaffre et Vanni 2020 ; Mayaffre et Vanni 2021a).

1.2. Liste de publications

Dans les chapitres suivants, l'ensemble des travaux présentés sont extraits de publications parues entre 2018 et 2021. Certaines contributions font néanmoins référence à des articles antérieurs, en effet cette thèse est le fruit d'une réflexion qui a débuté en 2013 au sein de l'UMR7320 Bases, Corpus, Langage (BCL) et qui se poursuit depuis en collaboration avec UMR7271 Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S). Une partie des résultats sont aussi extrait de publications à paraitre et illustre en partie les perspectives de ces travaux. À noter qu'un chapitre d'ouvrage intitulé "Deep learning et description des textes Architecture méthodologique" (figure 1.1) (Vanni et Precioso 2021) reprend en partie la structure des chapitres présentés ici et offre des prérequis méthodologiques (chapitre 2) et quelques études complémentaires concernant notamment la couche d'*Embedding* (section 2.5).

1.2.1. À paraitre

L. Vanni, M. Corneli, F. Precioso et D. Mayaffre (2022). « Text Class Activation Map (T-CAM) »

2. trois niveaux sont considérés : la forme graphique des mots, leur catégorie grammaticale et leur lemme

D. Mayaffre et L. Vanni (2021a). « Du texte profond. Textualité et deep learning »

1.2.2. 2018-2021

D. Mayaffre et L. Vanni (2021b). *L'intelligence artificielle des textes. Des algorithmes à l'interprétation.* Paris : Honoré Champion

L. Vanni et F Precioso (2021). « L'intelligence artificielle des textes ». In : Paris : Honoré Champion. Chap. Deep learning et description des textes Architecture méthodologique

E. Brunet, L. Lebart et L. Vanni (2021). « l'intelligence artificielle des textes ». In : Paris : Honoré Champion. Chap. Littérature et intelligence artificielle

S. Ruggia et L. Vanni (2021). « DeepFLE : la plateforme pour évaluer le niveau d un texte selon le CECRL ». In : *3 e Congrès Européen de la FIPF*. Athènes

L. Vanni, M. Corneli, D. Longrée, D. Mayaffre et F Precioso (2020b). « Key passages : from statistics to deep learning ». In : *Text Analytics. Advances and Challenges*, p. 41–54

D. Mayaffre et L. Vanni (2020). « Objectiver l'intertexte ? Emmanuel Macron, deep learning et statistique textuelle ». In : *15es Journées internationales d'Analyse statistique des Données Textuelles (JADT2020)*

L. Vanni, M. Corneli, D. Longrée, D. Mayaffre et F Precioso (2020a). « Hyperdeep : deep learning descriptif pour l'analyse de données textuelles ». In : *15es Journées internationales d'Analyse statistique des Données Textuelles (JADT)*

E. Brunet et L. Vanni (2019). « Deep learning et authentification des textes ». In : t. XXIV. 1. Textes et Cultures, Institut Ferdinand de Saussure, p. 1–34

L. Vanni, M. Ducoffe, D. Mayaffre, F Precioso, D. Longrée, V. Elango, N. Santos, J. Gonzalez, L. Galdo et C. Aguilar (2018). « Text Deconvolution Saliency (TDS) : a deep tool box for linguistic analysis ». In : *56th Annual Meeting of the Association for Computational Linguistics (ACL)*. Melbourne, p. 548–557

L. Vanni, D. Mayaffre et D. Longrée (2018). « ADT et deep learning, regards croisés. Phrases-clefs, motifs et nouveaux observables ». In : *14es Journées internationales d'Analyse statistique des Données Textuelles*. Rome, p. 459–466

1.2.3. Avant 2018

D. Mayaffre, C. Bouzereau, M. Ducoffe, M. Guaresi, F. Precioso et L. Vanni (2017). « Le vote disruptif. Les élections présidentielle et législatives de 2017 ». In : Paris : Presses SciencesPo. Chap. Les mots des candidats, de « allons » à « vertu », p. 129–152

M. Ducoffe, D. Mayaffre, F. Precioso, F. Lavigne, L. Vanni et A. Tre-Hardy (2016). « Machine Learning under the light of Phraseology expertise : use case of presidential speeches, De Gaulle -Hollande (1958-2016) ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles*

L. Vanni et A. Mittmann (2016). « Cooccurrences spécifiques et représentations graphiques, le nouveau "Thème" d'Hyperbase ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles*. T. 1, p. 295–305

L. Vanni, X. Luong et D. Mayaffre (2014). « Arbre et co-occurrences Nouvel outil logométrique sur le net. Application au discours de François Hollande ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles* 1, p. 639–649

Figure 1.1 – Ouvrage : L'intelligence artificielle des textes. Paris, Champion.

Chapitre 2

Prérequis

Ce chapitre reprend pour l'essentiel le chapitre d'ouvrage Vanni *et* Precioso *2021. Des modificiations mineures et certains enrichissements ont été apportés pour fixer les prérequis techniques nécéssaires à la comphénsion des autres chapitres*

2.1. Généralités

Longtemps présenté pour ses qualités prédictives, le *deep learning* montre aujourd'hui de nouveaux atouts et bénéfices pour l'Analyse de Données Textuelles (ADT). La boite noire qui existe entre les entrées et les sorties de l'IA, composée de couches de neurones artificiels, s'ouvre peu à peu et libère une connaissance inédite des textes. Ce grain à moudre supplémentaire profite à l'ADT dont les outils reposent depuis maintenant de nombreuses années sur des calculs statistiques stables et performants (Lebart, Pincemin et Poudat 2019). Le *deep learning* complète ainsi les méthodes traditionnelles de *l'école française d'analyse des données* (Beaudouin 2016) offrant un champ de vision plus large et un parcours interprétatif plus fin.

Nous proposons ainsi une méthodologie nouvelle développée à partir des réseaux de neurones artificiels. Ces réseaux sont construits à partir de différentes couches de neurones que l'on enchaîne les unes après les autres avec pour chacune des caractéristiques qui leurs sont propres. L'hypothèse de départ de notre contribution est que chaque type de réseau capture une partie de l'information linguistique qui oriente la prise de décision du modèle, dans le cas particulier d'une classification de texte. A terme, c'est cette information que nous nous proposons d'étudier, information potentiellement complémentaire, en tout cas comparable, à la statistique, et que nous appelons : nouveaux observables linguistiques.

Méthodologiquement, précisons encore que le point de départ de ces travaux est l'analyse statistique de données textuelles. Les outils tels que l'analyse factorielle des correspondances, le calcul des spécificités ou encore la classification ascendante hiérarchique sont les points de repère qui permettent d'identifier les plus-values descriptives de l'IA. Contraint par les exigences de l'ADT, nous avons développé une architecture particulière de *deep learning* qui répond aux besoins croissants des chercheurs face aux données à large échelle, les *big data*, qui semblent changer notre perception des textes, ou face à la *Distant Reading*

(Moretti 2013) qui semble changer nos pratiques séculaires de lecture.

Si les méthodes statistiques d'analyse du texte reposent pour la plupart sur des calculs historiques qui n'ont que peu évolué depuis des décennies, l'exploitation de ces méthodes a suivi l'évolution fulgurante de l'outil informatique. Comme nous l'avions remarqué en 2016 (Vanni et Mittmann 2016), le traitement de données textuelles à large échelle a ainsi engendré au fil du temps des *visualisations* nouvelles qui accompagnent de manière toujours plus performante les parcours interprétatifs des chercheurs désireux de prendre en main les *big data* textuelles.

Néanmoins certaines notions peinent à être capturées par nos outils statistiques classiques. En changeant d'approche, ne considérant plus les textes comme un sac de mots mais comme des ensembles structurés - notamment structurés linéairement - le *deep learning* propose un point de vue nouveau qui tente de répondre à certaines questions laissées par l'ADT en suspens.

Par exemple un type d'observations fortes qu'il faut attendre du *deep learning* concerne la notion de *passage*. Théorisé par (Rastier 2007), le *passage* est un morceau de texte, de taille variable, jugé suffisamment parlant pour être interprété. Il n'existe pas aujourd'hui d'outil statistique ADT capable de rendre compte du potentiel interprétatif d'un morceau de texte. Au mieux, les spécificités et les segments répétés peuvent attester de la distribution particulière des mots et des segments dans le corpus.

Avec les réseaux de neurones, il est possible d'observer des zones distinctives du texte plus ou moins longues, possiblement discontinues, définies par l'apprentissage du modèle. Les mots, les lemmes, les codes grammaticaux sont encodés en valeurs numériques et nous faisons l'hypothèse qu'un modèle entrainé recherche dans cette masse d'informations tout élément ou connexion d'éléments susceptibles de lui permettre de converger vers une solution optimale. Nous verrons plus loin que la nature et la complexité des éléments observés dépendent de la structure choisie pour le réseau et de son architecture.

Par extension, les *passages-clés* en ADT sont les passages les plus caractéristiques statistiquement d'un texte : ils ont été défini comme un segment qui concentre suffisamment de spécificités (Vanni, Mayaffre et D. Longrée 2018). Nous faisons l'hypothèse supplémentaire qu'un modèle entrainé à classer des textes sait identifier, dans le corpus, les passages les plus significatifs sur la base d'indices possiblement séquentiels, et plus proches de la qualité linguistique du texte.

De plus, les marqueurs linguistiques d'un texte combinent souvent plusieurs niveaux de représentation pour chaque mot (lexical, sémantique, syntaxique, ...). La complexité et la combinaison de ces marqueurs ont fait l'objet de nombreuses études qui se rapportent à la notion de *motif* proposée par (Mellet et D. Longrée 2009) : des observables lexico-grammaticaux complexes structurant et caractérisant un texte.

Un des enjeux majeurs du *deep learning* pour les SHS est donc de mettre au jour les

marqueurs qui ne seraient pas déjà connus, ni de la statistique textuelle ni peut-être de la philologie traditionnelle. Cet enjeu se double de la volonté d'expliquer les mécanismes internes (la boite noire) qui ont conduit à l'identification de ces marqueurs et d'en améliorer l'utilisation. Rappelons que l'objectif principal est bien de nourrir le parcours interprétatif du chercheur et non pas de se focaliser sur le score de prédiction du modèle. Combiner le fréquentiel et le séquentiel, considérer chaque mot dans un contexte local et global, croiser les différents niveaux grammatical et lexical des unités, voilà les paris que tente de relever le *deep learning* sur le texte.

2.2. Traitement des données textuelles

Les réseaux de neurones appliqués au texte peuvent avoir de multiples cas d'utilisation comme faire de la traduction automatique, générer du texte ou encore construire des classifieurs. Ce dernier cas est celui qui nous intéresse dans cette contribution.

La classification de textes automatique conduit les modèles à apprendre des caractéristiques utiles à la discrimination des classes. Pour obtenir un taux de précision élevé, ces modèles sont contraints (par optimisation) d'apprendre le plus possible de marqueurs significatifs de chaque classe.

En ADT, les classes définies par ce que l'on appelle communément les métadonnées sont le fondement des analyses statistiques contrastives. La qualité des descriptions dépend directement de la qualité du découpage du corpus (le partitionnement). Il en va de même pour le *deep learning* qui demande une certaine rigueur quant à l'homogénéité des données. Nous verrons plus loin dans ce chapitre que les biais d'un corpus qui sont détectables avec la statistique ont une influence négative forte sur la qualité d'un modèle entrainé. Une simple vérification des spécificités, une AFC [1] ou encore une distance intertextuelle cohérente sont des outils efficaces pour évaluer si les données sont suffisamment propres pour être exploitées par le *deep learning*. Le passage de la statistique au *deep learning* n'engendre pas de surcout de préparation des données si ce n'est une bonne maîtrise en amont de l'ADT classique.

Il reste un problème de taille. La quantité de données joue un rôle fondamental sur la qualité du modèle. Ici la statistique est moins sensible à ce phénomène et quelques milliers de mots sont souvent suffisants pour obtenir de premiers résultats satisfaisants. Or le *deep learning* fonctionne par *apprentissage* (on ne fait pas d'hypothèse sur un modèle sous-jacent, on va le construire). Chaque donnée analysée entraine un succès ou un échec du modèle qui doit corriger sa représentation des données pour améliorer ses performances, on dit que l'apprentissage est supervisé. Le mécanisme responsable de cette correction s'appelle la *backpropagation* (rétropropagation du gradient en français), et demande une quantité de données importante pour permettre au réseau d'atteindre un état stable et un

1. L'analyse factorielle des correspondances (AFC) est une méthode statistique qui permet d'analyser les relations qui peuvent exister entre des individus et des variables placées dans un tableau rectangulaire de données, autrement appelé tableau de contingence. Cette méthode, développée par Jean-Paul Benzécri dans les années 1960, comparable à l'analyse en composantes principales (ACP), vise à projeter sur des axes les données de façon que l'on puisse en visualiser et analyser les corrélations sur un graphique.

taux de précision acceptable. Pour comprendre ce phénomène un rapide détour théorique est nécessaire sur le fonctionnement des réseaux de neurones artificiels et l'historique modèle du *perceptron* (Rosenblatt 1958).

Le texte, une fois converti en valeurs numériques (voir *Embedding* section 2.5), est le point de départ de l'ajustement des paramètres du *deep learning* (ou plutôt le point d'arrivée si on parle de *backpropagation*). Chaque mot représente un *état d'activation* d'un ou plusieurs neurones en entrée du réseau. Ces neurones communiquent ensuite avec l'ensemble des autres neurones de la couche suivante en transmettant cette information. Le mécanisme général de base est le suivant : chaque état y d'un neurone de la couche suivante est calculé en fonction de la somme des activations reçues de la couche précédente x_i pondérées par un paramètre w_i pour chaque neurone. Cette somme est ensuite ajustée par une fonction φ d'activation qui détermine la valeur finale retenue (l'état d'activation) de sortie du neurone (figure 2.1) .

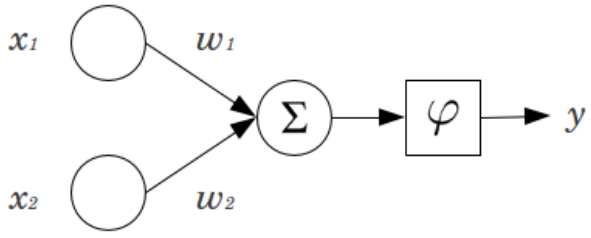


Figure 2.1 – Modèle standard du perceptron : calcul du taux d'activation de chaque neurone

Lorsque cette transmission d'informations ne va que de la couche d'entrée vers la couche de sortie (en passant au travers des différentes couches du réseau) on dit que le réseau est *feed forward*. L'ensemble des valeurs des paramètres w_i représentent l'état global du modèle que le réseau doit ajuster pour améliorer ses performances. Pour ce faire, l'algorithme de *backpropagation* consiste en un calcul du gradient de l'erreur pour chaque neurone en partant de la dernière couche (la sortie) jusqu'à la première (l'entrée), c'est-à-dire un ajustement mathématique des pondérations w_i dans le sens d'une diminution de l'écart entre la sortie calculée et celle attendue (que le réseau doit apprendre).

Dans le cas trivial de problèmes dits linéairement séparables, une seule couche de neurones suffit pour trouver une solution en un nombre fini de cycles d'apprentissage. Avec une couche supplémentaire (une couche cachée), le réseau commence à résoudre des problèmes plus complexes. Certains réseaux particuliers se comportent comme nos outils d'ADT, c'est le cas par exemple des réseaux de type *auto-encoder*[2] et de l'analyse factorielle des correspondances (Lebart 1997). Mais dans la majorité des cas, de nombreuses couches cachées sont nécessaires, utilisant des architectures particulières qui augmentent considérablement le nombre de paramètres w_i à calculer. L'ajustement de l'ensemble de ces paramètres entraine un cout d'exécution important et sollicite une quantité de données

2. Les *auto-encoders* sont des réseaux de neurones dont la couche d'entrée est similaire à la couche de sortie, la couche intermédiaire (la couche cachée) ayant pour rôle d'encoder ou de compresser l'information sous forme vectorielle.

considérable.

En pratique ce mécanisme se traduit donc par des tailles de corpus requises importantes mais aussi variables selon la difficulté de la tâche d'apprentissage demandée à la machine. La taille minimum de chaque partie (chaque classe) peut varier d'une centaine de milliers de mots pour des tâches simples à plusieurs millions mots pour les tâches les plus complexes. Il n'y a pas de règle absolue, mais en général il faut garder à l'esprit que les performances du modèle généré sont directement corrélées au volume de données disponibles pour l'apprentissage. Cette sensibilité à la taille des données est une des restrictions fortes de l'usage du *deep learning* pour le traitement des textes. Avant de se tourner vers ce type de méthode il est donc nécessaire de vérifier si la quantité de données dont on dispose est suffisante pour entrainer un réseau de neurones profond.

2.3. Architecture deep learning

L'architecture générale du modèle proposé ici pour la linguistique textuelle est une extension de l'état de l'art en termes de classification de textes. Pour comprendre les différences et les ajouts opérés il est nécessaire d'appréhender tout d'abord le modèle standard de classification. Il est composée de plusieurs composants principaux successifs : un *Embedding*, un réseau *Convolutionnel* (CNN) associé à une couche de *Pooling* (regroupement) et un réseau *Dense* (MLP) (figure 2.2). Chacun de ces composants est connecté au suivant et est responsable d'une partie précise de l'apprentissage.

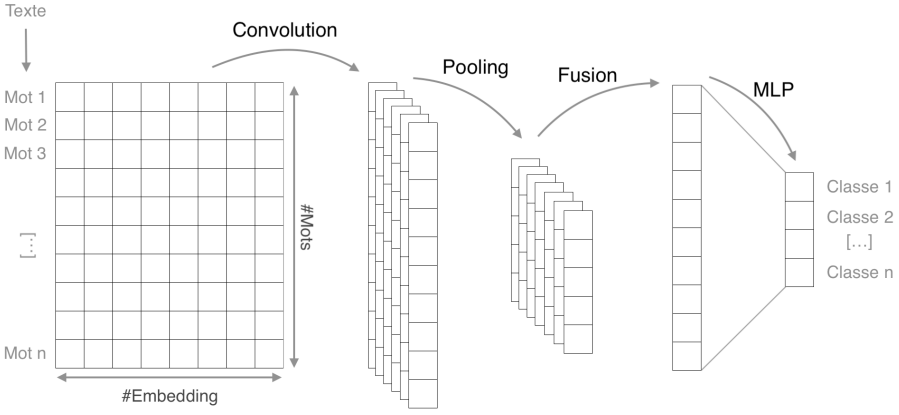


FIGURE 2.2 – Architecture générale pour la classification de texte

2.3.1. Représentation des mots

La méthode la plus simple pour représenter un mot dans un réseau de neurones est de l'encoder par un identifiant unique, un entier. Cette méthode ne prend pas en compte le sens ontologique des mots. Dès lors, le rôle de l'embedding est d'associer à chaque mot un vecteur (potentiellement de grande dimension) de telle manière que la distance euclidienne des vecteurs reflète, sinon directement le sens des mots, toutefois leurs rapports et organisation sémantiques (figure 2.3).
Cette représentation est devenue ces dernières années un standard qui permet d'améliorer

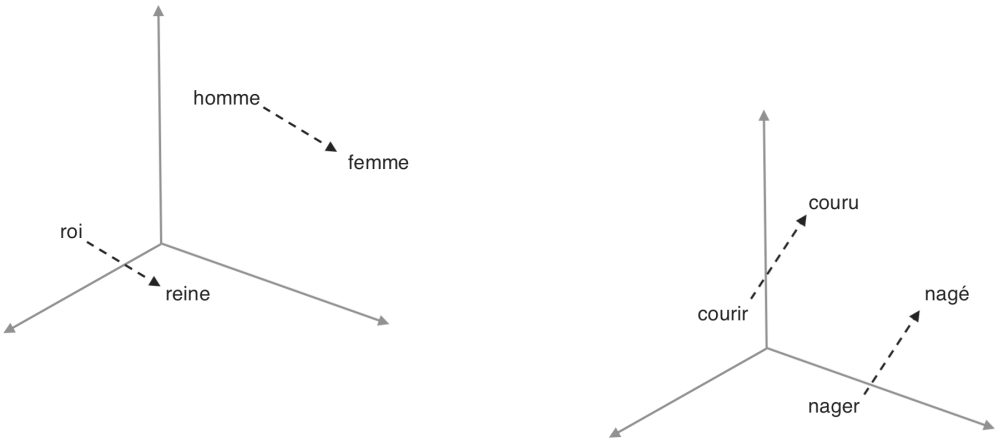


FIGURE 2.3 – *Embedding* ou espace vectoriel des mots

considérablement les performances en tâches de classification ou de traduction de textes (R. COLLOBERT et WESTON 2008). Ces méthodes qui basent la représentation des mots sur leurs contextes d'emploi locaux (les quelques mots qui les précèdent et les suivent, les constructions syntagmatiques dans lesquelles ils se trouvent) permettent de capturer la valeur syntaxique et sémantique de chaque mot dans le corpus pris comme modèle. Leur construction fait généralement appel, elle-même, à un réseau de neurones (avec une structure particulière, cf. figure 2.7 plus loin), mais les méthodes statistiques de l'analyse des données sont susceptibles de définir ces espaces vectoriels (AFC, LSA, etc.) (cf. section 2.5). Parmi les représentations déjà calculées et disponibles les plus célèbres, il y a *Word2Vec* (T. MIKOLOV et al. 2013) et *Global Vectors* (GloVe) (PENNINGTON, SOCHER et C. MANNING 2014), ou plus récemment *BERT* pour l'anglais et *FlauBERT* ou *CamemBERT* pour le français, qui proposent tous des représentations vectorielles préentrainées sur de très larges corpus (plusieurs milliards de mots).

De telles représentations peuvent alors être mises en œuvre dans le réseau de neurones ici utilisé pour classer les textes soit en se basant sur un préapprentissage déjà fait (en reprenant la représentation *Word2Vec* par exemple) soit en construisant la représentation des mots au fil de l'apprentissage de la tâche de classification. Dans tous les cas, l'embedding est un paramètre du réseau qui peut être corrigé à tout moment au fil de l'apprentissage. La représentation des mots est un élément princeps déterminant dans la compréhension de la prise de décision du réseau. Nous verrons cela en détail dans le chapitre 2.5.

2.3.2. Representation des textes

Les réseaux de neurones convolutionnels, CNN (LECUN et al. 1998), sont des réseaux type *feedforward* avec des couches de convolution et des couches de regroupement (*Pooling*) à deux dimensions. Inspirés du cortex visuel et initialement développés pour le traitement d'images, ils utilisent un mécanisme de filtres qui balayent l'information (l'image) de gauche à droite et de haut en bas, pixel par pixel, pour ne retenir que les parties principales et pertinentes de l'image. Encouragés par les résultats obtenus sur les images, ces

réseaux à convolution ont été adaptés au texte (Collobert et al. 2011 ; Kalchbrenner, Grefenstette et Blunsom 2014 ; Kim 2014a). Avec les données textuelles, l'entrée du réseau est un tableau unidimensionnel représentant numériquement les séquences de mots étudiés [3]. Ici l'architecture des CNN standard est légèrement modifiée pour s'adapter au caractère unidimensionnel du texte (linéaire de gauche à droite) et effectuer des opérations de convolution et de regroupement à une dimension (du premier mot au dernier). Il existe de nombreuses variations de ce modèle comme celui de R. Collobert et Weston 2008 qui a proposé différentes architectures pour prendre en compte certains aspects du texte comme l'analyse morphosyntaxique, la reconnaissance d'entités nommées ou encore l'étiquetage sémantique.

Ce type de réseaux de neurones considère donc les données textuelles comme des images avec des caractéristiques statiques et spatiales (segments répétés, collocations). Cette approche contraint le type de données en entrée du réseau avec des tableaux d'entiers de taille fixe. Chaque texte doit donc être découpé en amont en séquences de taille fixe pour pouvoir être ensuite analysé [4] . Il existe plusieurs méthodes de découpage.

Les signes de ponctuation forte sont souvent utilisés pour délimiter un morceau de texte significatif ; *grosso modo* des phrases. Afin de normaliser, une phrase trop longue est simplement coupée et une phrase trop courte est complétée par une suite de mots arbitraire notée *PAD* (pour *padding*). Cette solution fait partie de l'état de l'art en termes de découpage de texte pour le *deep learning*. Mais comme nous l'avons vu en introduction, la détection de passages significatifs dans un texte pousse à s'affranchir des seuls marqueurs typographiques qui pourraient masquer certains marqueurs importants dans une classe. Il a donc été décidé ici de couper le texte en utilisant une fenêtre semi-coulissante tout en conservant la ponctuation qui devient un mot parmi les autres et non plus un délimiteur du passage.

Ce découpage du texte a deux effets significatifs : d'une part il laisse au *deep learning* la possibilité de détecter lui-même les empans linguistiques qui seront déterminants pour la décision finale du réseau (Vanni, Mayaffre et D. Longrée 2018), d'autre part il permet de diviser la prédiction globale d'un texte long en plusieurs sous-prédictions de textes plus courts et ainsi permettre une première réflexion sur la topologie du texte décrite par le *deep learning* (figure 2.4).

L'architecture générale d'un réseau dépend donc des tâches que l'on demande à l'IA. De nombreuses couches successives peuvent être emboitées les unes dans les autres pour former une chaîne de traitement particulière qui répond à un besoin précis. Mais il n'y a pas que l'architecture globale qui donne ses qualités à un modèle. La configuration fine de chaque élément du réseau, ce qu'on appelle les *hyperparamètres*, détermine aussi fortement la pertinence finale du modèle entrainé.

3. En fait, avec un *Embedding*, chaque mot est aussi représenté par une ligne qui correspond à la taille de l'*Embedding*, mais l'important ici est le nombre de lignes qui correspond au nombre de mots dans la séquence.

4. Le lecteur a compris qu'il s'agit de séquences auxquelles on va chercher à attribuer une classe.

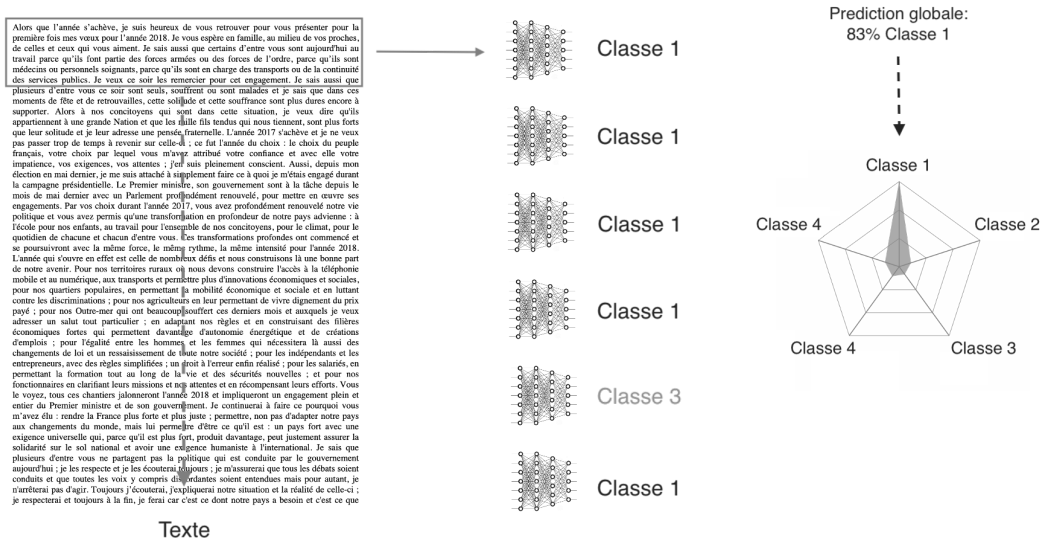


FIGURE 2.4 – Fenêtre coulissante et découpage du texte

2.4. Paramètres et hyperparamètres

Nous avons vu que le système d'apprentissage du *deep learning* repose sur des paramètres w_i estimer. Ce sont les poids de chaque neurone du réseau. Ces poids sont calculés automatiquement pendant la phase d'apprentissage et sont la représentation numérique des nouveaux observables que nous allons examiner dans les chaptitres 3, 4 et 6. Mais une architecture *deep learning* repose aussi sur d'autres paramètres que le réseau ne contrôle pas pendant la phase d'apprentissage. Ces paramètres concernent la structure même du réseau et modifient potentiellement l'efficacité de chaque itération de l'entrainement (*Epoch*[5]). Ce sont les *hyperparamètres* du modèle.

Il existe de nombreux *hyperparamètres*, tous ne sont pas nécessaires à la compréhension du *deep learning* pour l'ADT. Nous allons détailler dans les sous-sections suivantes ceux qui ont un impact majeur sur l'interprétabilité des résultats et ceux qui modifient significativement la qualité d'un modèle en fonction des données.

Les observations seront appuyées ici sur un indicateur donné par le réseau à chaque itération à savoir le taux de précision (*accuracy*), qui correspond à la précision du modèle en pourcentage. Ce taux est calculé automatiquement en utilisant les données réservées à la validation qui correspondent à un certain pourcentage de l'ensemble. Ce pourcentage est d'ailleurs certainement un des premiers hyperparamètres du réseau que l'on manipule. En général il varie entre 10% et 20%, mais dépend directement de la taille des données disponibles pour l'apprentissage. Plus l'ensemble de validation est grand plus le score d'*accuracy* sera précis, mais moins le modèle aura de données d'entrainement pour améliorer ses performances. À noter qu'un autre indicateur que le taux de précision, la

5. L'apprentissage demande plusieurs passages sur l'ensemble des données d'entraînement que l'on appelle *Epoch*.

fonction de perte (*loss function*), est utilisé pour enregistrer le comportement du réseau et indique si un modèle devient trop précis sur les données d'entrainement et donc devient incapable de généraliser sur de nouvelles données, on parle ici d'overfitting. Plus le *loss* est bas plus le modèle est capable de s'adapter. Au cours de l'apprentissage, il est fréquent de voir cette valeur baisser pour atteindre un seuil minimum puis augmenter de nouveau lorsque le modèle commence le surapprentissage. Il convient alors de stopper l'enregistrement des paramètres. Cette méthode s'appelle le *Early-stopping* (Prechelt 1998) et suit en général l'exemple de courbe présenté dans la figure 2.5.

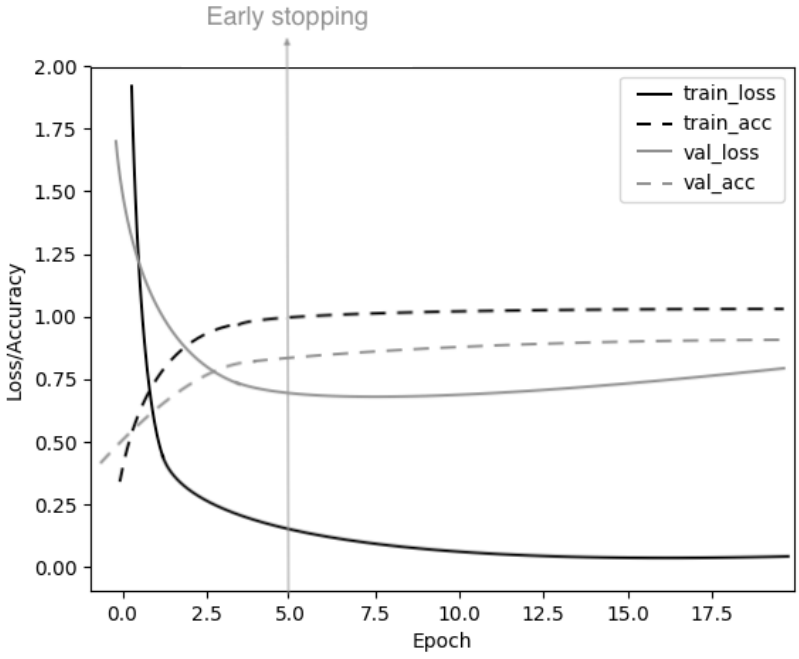


Figure 2.5 – Arrêt précoce de l'apprentissage, *Early-stopping*

Il existe donc plusieurs types de paramètres qui impactent différemment le réseau. Certains sont purement d'ordre computationnel (comme le nombre d'itérations sur les données, le nombre de couches cachées, ou le nombre de neurones dans chacune de ces couches) et d'autres en revanche trouvent des explications directement linguistiques jusqu'à nous faire mieux comprendre les textes.

2.4.1. Enjeux linguistiques

La convolution repose sur plusieurs hyperparamètres dont deux fondamentaux qu'il est nécessaire d'analyser pour pouvoir interpréter les résultats et mesurer la plus-value descriptive du *deep learning*. Le premier paramètre ne concerne pas directement l'architecture du réseau de neurones, mais plutôt la structure des données en entrée du réseau. C'est la **taille du segment de texte à classer** : la séquence de mots qui est donnée au modèle pour apprendre ou prédire une classe. Cette séquence a une taille fixe qu'il faut raisonnablement évaluer en fonction du corpus que l'on utilise et de l'hypothèse de travail qu'on

souhaite tester. Avec l'architecture qui est proposée ici, la taille est fonction du nombre de tokens (*grosso modo* les mots-occurrences) dans le texte. En ADT un token est typiquement une chaîne de caractères entre deux caractères séparateurs (espace, ponctuation). Il faut alors trouver un ratio convenable entre le nombre de séquences que l'on peut extraire d'un corpus et la taille de ces séquences. En effet le nombre de séquences n disponible pour un entrainement peut être évalué par la taille du corpus T divisé par la taille de chaque séquence t : $n = \frac{T}{t}$ [6].

Plus un corpus est volumineux plus il sera possible de le découper en n séquences donc d'augmenter la taille de ces séquences. Une séquence plus longue donnera plus de matière au réseau pour apprendre à chaque essai. Mais il faudra alors aussi nécessairement plus de séquences puisque cette matière correspond directement à de nouveaux paramètres w_i qu'il faudra estimer lors de l'apprentissage. Il faut donc trouver un juste milieu en surveillant l'impact sur l'*accuracy* et le *loss* à chaque nouveau modèle entrainé.
Voici un exemple de mesures sur un corpus qui regroupe l'ensemble des discours des 8 présidents français de Ch. de Gaulle à E. Macron, soit huit classes au total pour près de 3 millions de mots (tableau 2.1) :

Taille séquence	20	50	100	200	400
Nombre de séquences	110 000	44 000	22 000	11 000	5500
Accuracy (%)	77.5	85.5	92.51	**96.13**	94.17
Loss	69.2	45.71	21.51	**11.73**	18.09

TABLE 2.1 – Comparaison de l'*accuracy* et du *loss* avec des différentes tailles de séquence

On constate ici que la précision du modèle augmente proportionnellement à la taille de la fenêtre avant de reculer lorsque le nombre de séquences devient insuffisant.

Avec un peu d'habitude et une connaissance approfondie des données que l'on entraine, on peut ajuster intuitivement ce paramètre et optimiser le nombre d'essais nécessaire. Du point de vue de l'ADT, ce résultat encourage aussi à revoir certains de nos calculs statistiques, comme le calcul de la cooccurrence par exemple qui peut s'affranchir facilement de la ponctuation ou du retour charriot pour y préférer une fenêtre de taille fixe plus large (par exemple 200 mots ici). En général cet aller-retour entre statistique et *deep learning* profite à chacune des deux approches. (BRUNET, LEBART et VANNI 2021) montre un cas particulier qui permet d'améliorer significativement les résultats du calcul statistique de la distance intertextuelle en prenant en compte certaines observations faites avec le *deep learning*.

D'autres paramètres sollicitent davantage encore nos connaissances linguistiques du corpus. Le deuxième hyperparamètre de la convolution étudié ici est la taille des empans glissants qui capturent les contextes de mots. L'idée de la convolution est de balayer le texte avec des filtres de différentes tailles pour capturer les saillances du texte sur des fenêtres de plusieurs mots. Avec cette méthode chaque mot est ainsi considéré dans son

6. En réalité il faut ajouter un facteur 2 si on opte pour la stratégie de la fenêtre semi-coulissante.

contexte proche. Soit la phrase « je pense donc je suis ». Chaque mot graphique peut être représenté soit par une valeur unique comme en ADT, soit par une valeur composée du contexte immédiat comme le propose la convolution (figure 2.6).

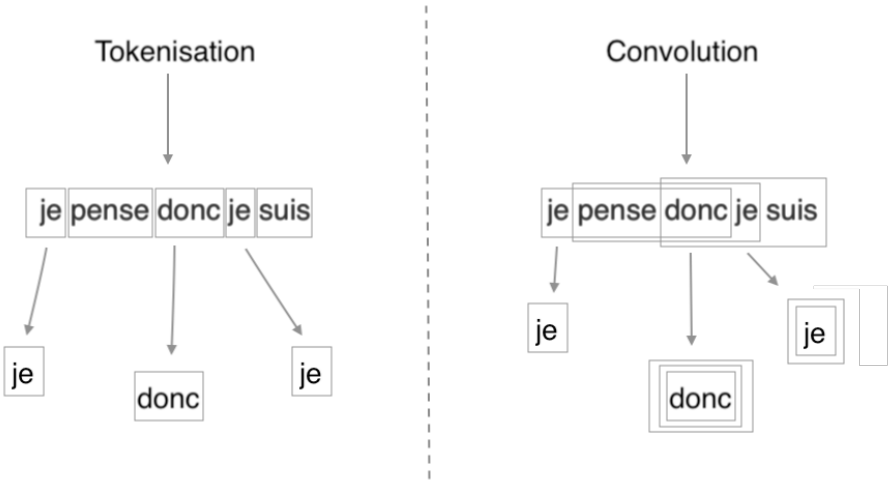


FIGURE 2.6 – Représentation des mots en ADT et en deep learning

L'approche fréquentielle statistique classique donne aux mots une valeur commune pour toutes ses occurrences, quels que soient ses contextes. Cette représentation pourra utiliser plusieurs approches (la distribtuion, la cooccurrence, le z-score, etc.) mais elle est unique : ici, dans la partie gauche de la figure, les deux occurrences du mot *je* ont bien la même représentation.

Avec la convolution, l'approche passe du fréquentiel au séquentiel, chaque mot est pris dans son contexte et engendre une représentation vectorielle particulière liée au contexte. Même si le fréquentiel joue un rôle important dans l'apprentissage, cette fenêtre contextuelle coulissante garantit une analyse séquentielle et contextualisante des mots. En pareil cas, partie droite de la figure, le mot *je* a donc deux représentations différentes, une en début de séquence lorsque le *je* précède *pense* et *donc* et une au milieu entouré de *donc* et *suis*. L'importance de la taille des filtres de la convolution dans le paramétrage du réseau semble alors évidente. Le tableau 2.2 montre un exemple de sortie machine qui illustre ce comportement où f est la taille de la fenêtre choisie.

Dans une vaste étude sur l'évolution du discours du Front National en France (BOUZEREAU 2020), les effets de ce paramètre sur l'affiliation du discours de Marine Le Pen de 2017 ont été comparés dans les différentes campagnes présidentielles de 2007 à 2017. La tâche demandée au réseau est d'affilier le discours de la représentante du FN en 2017 à l'un des candidats de l'une des campagnes. Outre la meilleure compréhension de l'*intertexte* (voir chapitre 7) grâce au *deep learning*, cette étude montre ici que la taille la fenêtre de convolution de 3, 6 ou 9 mots, notée f dans le tableau, modifie significativement l'attribution du discours. Le discours emprunte visiblement d'une part au discours fondateur du

f	2007 (%)			2012 (%)			2017 (%)		
	Roy	Sarko	LePen	Holl	Sarko	M.LePen	Melench	Macr	Fill
3	6,46	13,04	**21,67**	2,07	5,64	19,08	11,37	4,46	16,22
6	11,3	11,82	14,17	3,33	4,54	**21,8**	10,21	5,38	17,44
9	8,57	**24,97**	9,42	3,69	5,06	13,21	10,88	3,99	**20,22**

TABLE 2.2 – Effet de la taille de convolution sur l'attribution de segments de discours de Marine Le Pen de 2017 à différents discours de campagne de 2007 à 2017

FN incarné par J.-M. Le Pen en 2007 et d'autre part à la rhétorique de la droite traditionnelle de Sarkozy la même année ou de Fillon en 2017. Entre lexique et phraséologie, la taille de la convolution pousse visiblement le réseau à se concentrer sur des marqueurs différents. Une taille réduite conduira nécessairement le réseau à se focaliser sur du lexique et de la spécificité de mots, alors qu'une taille plus large permettra au réseau de détecter des motifs linguistiques plus étendus mais plus rares qui pourraient caractériser plus précisément un locuteur. On observe donc sur cet exemple que le discours de M. Le Pen en 2017 est composé du matériau lexical historique du discours du FN mais que ce matériau est combiné en enchaînements phraséologiques proches de la droite classique.

Il existe d'autres *hyperparamètres* liés à la *convolution*, comme par exemple le nombre de filtres utilisés. Ces paramètres participent principalement à définir la profondeur du réseau et sont beaucoup moins évidents à interpréter. Chaque composant d'une architecture *deep learning propose* au moins un paramètre de ce type qui joue sur la taille du réseau. Même si elle n'offre pas de plus-value descriptive directe, nous allons voir maintenant que la manipulation de ces autres *hyperparamètres* est importante puisqu'elle permet au modèle de converger vers un taux de précision élevé (en classification), le prérequis nécessaire à notre démarche d'analyse des couches cachées du *deep learning*.

2.4.2. Enjeux informatiques

Parmi les autres hyperparamètres que propose le *deep learning*, nombreux sont ceux qui affectent directement le nombre de neurones ou le nombre de couches cachées du réseau pour augmenter ou diminuer la complexité du modèle et sa capacité à accomplir des tâches.

Au niveau de l'architecture, il est possible par exemple de modifier la taille du vecteur qui représente les mots en entrée (taille de l'embedding) ou encore, modifier la taille et le nombre de couches cachées dans la partie dense du réseau. Pour tous ces hyperparamètres, il n'y a pas de valeur idéale par défaut. Dans le cas de l'embedding, le modèle doit simplement avoir suffisamment de dimensions pour bien discriminer les mots. Mais comme avec la plupart des hyperparamètres *deep learning*, il faut trouver un juste milieu. Un vecteur trop grand conduit le modèle au surapprentissage. Et un vecteur trop petit ne permet pas de converger vers une solution optimale. À noter que plus le nombre de neurones est élevé dans un réseau (c.-à-d. nombre de couches, filtres, embedding ...) plus le nombre de paramètres w_i à estimer est élevé et donc le temps de calcul aussi.
La *backpropagation* propose aussi son lot d'hyperparamètres. Le premier est le *learning rate* qui joue sur l'amplitude de l'ajustement de chaque paramètre w_i pendant l'apprentis-

sage. Une amplitude élevée permettra au réseau de progresser plus vite (moins d'itérations sont nécessaires), mais la solution optimale (le taux d'*accuray* le plus élevé) ne sera peut-être jamais atteinte et le réseau risque d'osciller autour en modifiant trop grossièrement les paramètres. En revanche un *learning rate* petit garantira une progression lente et efficace tout au long de l'apprentissage, mais le nombre d'itérations nécessaires peut devenir rapidement très élevé et ralentir considérablement la phase d'apprentissage. En général cette valeur oscille entre 0.001 et 0,0001.

Pour finir, le nombre d'*epochs* et la taille de *batch* permettent au réseau d'établir une stratégie pour optimiser le temps de calcul. Ils sont en général étroitement liés à l'architecture de la machine sur laquelle est exécuté l'apprentissage. La taille du *batch* correspond au nombre d'exemples que le réseau utilise en même temps pour corriger les poids w_i de ses neurones. Plus il y a d'exemples testés en même temps plus le réseau aura de matière pour ajuster ses paramètres. Les paramètres sont ainsi calculés à chaque *batch* (lot d'exemples) et une *epoch* correspond à un passage complet sur l'ensemble des données. Une taille de *batch* élevée permettra un parcours plus rapide de l'ensemble des données, mais entrainera un nombre de calculs numériques plus élevé pour la descente de gradient.

C'est ici que l'architecture de la machine joue un rôle important. Il n'y a pas de *deep learning* sans évoquer l'utilisation de cartes graphiques surpuissantes. En effet en informatique, tous les processeurs ne sont pas équivalents en termes de performances par rapport aux tâches qui leur sont demandées. Sans entrer dans les détails, il existe des processeurs de type CPU (*Central Processing Unit*) idéaux pour effectuer les tâches séquentielles de nos activités de bureautique standards, et les processeurs de type GPU (*Graphics Processing Unit*) conçus pour le calcul numérique et le calcul parallèle, idéaux donc pour notre descente de gradient sur de multiples exemples. Une machine équipée d'un processeur de type GPU acceptera donc plus facilement des tailles de *batch* élevées et chaque *batch* sera parcouru très rapidement (temps d'apprentissage réduit). Mais attention, comme l'ensemble des hyperparamètres du *deep learning*, une taille de *batch* trop élevée dégradera la qualité de l'apprentissage puisqu'il y aura trop d'exemples à prendre en compte à chaque *backpropagation* pour pouvoir trouver la solution optimale. Enfin, le nombre d'itérations sur l'ensemble des données (*epoch*) peut être ajusté pour prolonger l'entrainement jusqu'à atteindre une précision *acccuracy* convenable. En général ces deux paramètres sont corrélés, on augmente la taille du *batch* en même temps que le nombre d'*epoch* car l'apprentissage est plus rapide, mais il nécessite plus d'itérations sur l'ensemble des données.

Les *hyperparamètres* sont donc essentiels à la fabrication d'un modèle robuste et pertinent pour l'ADT. Il existe de nombreux autres *hyperparmètres* qui ne seront pas abordés dans ce chapitre, comme le type de fonction d'activation de chaque neurone ou encore les différents *optimizer* qui permettent de corriger l'erreur du modèle. Chacun peut influencer la qualité de l'apprentissage. Une bonne pratique suggérée en général en *deep learning* est de réutiliser des paramètres fixés sur un modèle proche de celui que l'on souhaite entrainer et de tester ensuite pas à pas l'effet de chacun d'eux sur le comportement du modèle global.

Si la prise en main de ces nouveaux outils demande donc un certain temps d'adaptation et de compréhension, les promesses du *deep learning* descriptif sont grandes. Nous allons voir maintenant en quoi les réseaux de neurones pourraient répondre aux attentes des arts et des sciences du texte.

2.5. Préappentissage des mots

Il existe plusieurs façons de représenter un texte en entrée d'un réseau de neurones. Pour des raisons descriptives et par souci d'interprétabilité, il est efficace de choisir une représentation au niveau du mot. Les caractéristiques d'un texte semblent plus évidentes à interpréter à partir des mots qu'à partir des lettres. De plus, cette représentation se situe sur un pied d'égalité avec l'ADT qui représente aussi le texte comme une suite de tokens. Cette représentation des mots passe par leur conversion numérique sous forme de vecteurs qu'on appelle *Embedding*. Chaque dimension de ce vecteur est un des paramètres w_i que le réseau doit estimer au cours de l'apprentissage.

Pour obtenir une représentation pertinente des mots, une des solutions les mieux reconnues par la communauté est d'utiliser une méthode appelée *Word2Vec*, sorte de préapprentissage qui utilise différents modèles possibles comme *CBOW* ou *Skip-Gram* de (T. Mikolov et al. 2013). Cette méthode efficace prend en compte l'axe syntagmatique et l'axe paradigmatique en proposant une représentation en contexte des mots, en utilisant un réseau de neurones pour prédire les cooccurrences spécifiques d'un corpus d'entrainement (figure 2.7). La couche cachée représente alors l'*embedding* du mot et le nombre de neurones utilisés dans cette couche correspond à la dimension du vecteur. Ce modèle donne un moyen efficace de fixer la représentation des mots avec un état compréhensible pour l'homme basé sur la cooccurrence des mots.

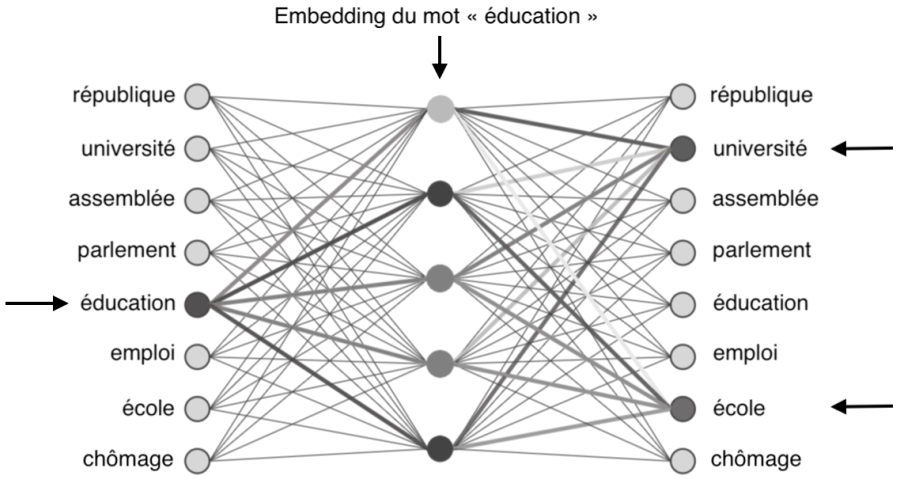


Figure 2.7 – Model de préapprentissage de l'embedding basé sur le modèle *SkipGram*. Capture faite à partir de l'application `http://ronxin.github.io/wevi/`

En ADT, la cooccurrence est une approche bien connue pour décrire les tokens et leur

restituer leurs charges sémantiques. Il existe par exemple une méthode implémentée par le logiciel Hyperbase (voir section 8.1), *corrélat*, qui vectorise chaque mot sur la base de ses cooccurrents. Pour capturer les thèmes d'un corpus, *corrélat* construit une matrice carrée du vocabulaire (matrice Mot x Mot) qui représente la fréquence de chaque cooccurrence en valeur absolue (ou en écart réduit). Cette représentation est suffisamment robuste pour pouvoir ensuite visualiser avec une AFC les principaux thèmes d'un corpus. Par exemple, la figure 8 présente les 300 substantifs les plus fréquents du discours présidentiel français. On observe rapidement avec cette méthode ici les principale *isotropie* (Viprey 2006) du discours politique. On retrouve ainsi dans les 4 quadrants du graphique l'économie, la politique intérieure, la politique sociale et la politique internationale.

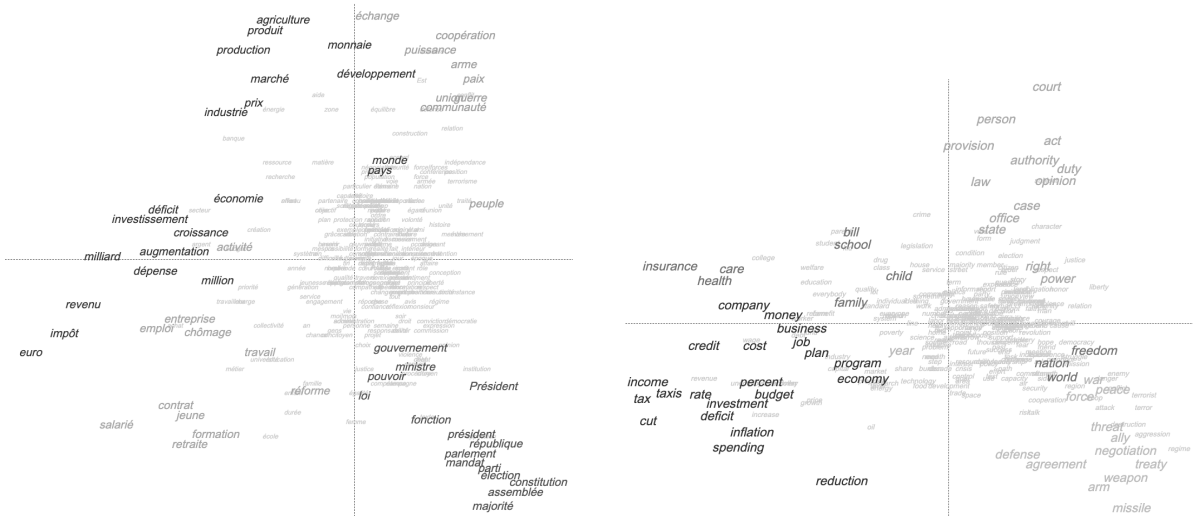


Figure 2.8 – Analyse de l'espace vectoriel des cooccurrences (calcul statistique).

Cette représentation des mots par leurs profils cooccurrentiels est exactement ce que l'on obtient avec *Word2Vec*. La couche cachée du réseau de neurones donne un vecteur qui représente les profils cooccurrentiels. Pour le vérifier on peut projeter avec une AFC les mêmes 300 substantifs les plus fréquents que précédemment, mais en utilisant cette fois le profil généré par *Word2Vec*. (figure 2.9).

Le constat est évident, les mêmes observations que celles obtenues avec le calcul des cooccurrences statistiques classiques sont possibles. Le réseau de neurones sensible à la cooccurrence crée un espace vectoriel des mots finalement proche de celui fabriqué par l'ADT. Ce phénomène est directement lié aux résultats proposés par Lebart 1997. Les réseaux de neurones utilisés pour fabriquer ces représentations vectorielles sont aussi appelés des *auto-encodeurs*, sorte de réseau particulier avec une seule couche cachée où les entrées et les sorties correspondent à des mots. Il a été montré que ces réseaux particuliers se comportent comme l'analyse factorielle des correspondances (AFC) avec les lignes et les colonnes du tableau de contingence qui correspondent respectivement aux entrées et sorties du réseau. Même si ce cas particulier implique des fonctions d'activations linéaires,

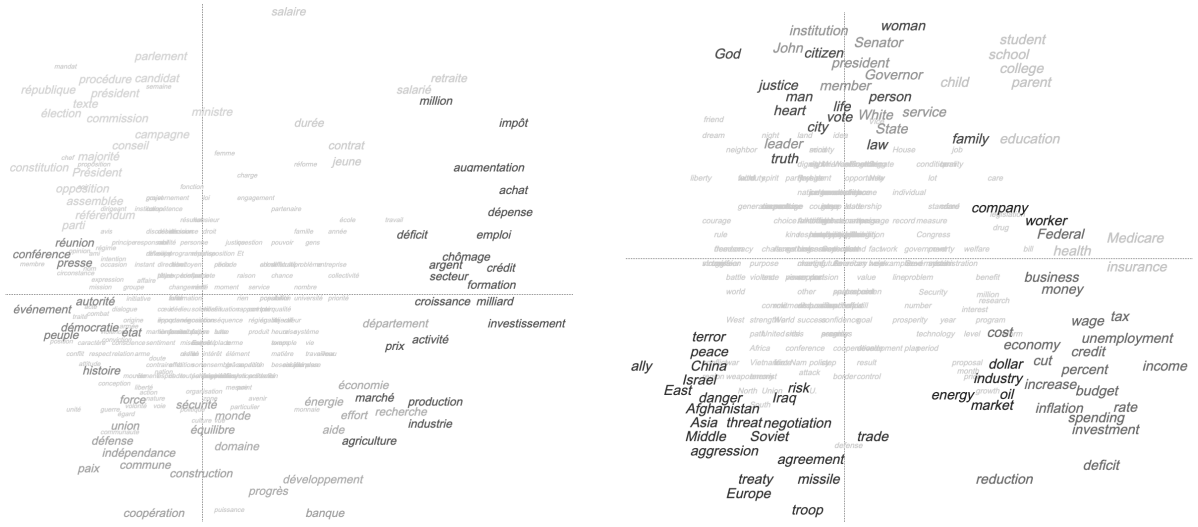


FIGURE 2.9 – AFC du tableau des mots décrits par leurs coordonnées issues de l'embedding calculé en utilisant *Word2Vec* (*SkipGram*) sur le corpus des présidents français

les deux approches convergent vers le même type de représentation. Le préapprentissage de *Word2Vec*, couteux en termes de calcul, peut donc être remplacé de manière évidente par nos outils d'ADT classiques pour générer des représentations de mots sensibles aux associations sémantiques. Néanmoins, il y a un phénomène qui doit attirer notre attention et qui pourrait bien redonner l'avantage aux réseaux de neurones.

Lorsque l'on entraine un modèle *deep learning* pour une tâche de classification précise, on permet en général au réseau de corriger cette représentation au cours de l'apprentissage pour améliorer les scores de précision (*backpropagation*). Le modèle semble donc devoir ajuster cette représentation pour pouvoir converger vers une solution optimale. Cela veut dire que les représentations basées sur les cooccurrences utilisées en entrée du réseau (contextes locaux autrement appelés micro-distribution des mots dans le texte) ne sont pas forcément les plus pertinentes pour la classification des textes demandée. La recherche de la performance perturbe ainsi les certitudes générales sur l'*embedding* des mots.

En effet, si on regarde les poids des neurones représentant l'espace vectoriel des mots après l'apprentissage, on observe qu'ils sont largement modifiés par rapport à ceux d'origine. L'expérience de l'AFC sur ce nouvel *embedding* révèle une cartographie lexicale tout à fait nouvelle (figure 2.10).

Difficile d'interpréter ici les pôles des axes principaux de cette AFC. Les thèmes récurrents dans le discours sont visiblement moins nets. Il semble que la cooccurrence ait finalement assez peu d'importance pour fabriquer un *embedding* pertinent pour la tâche de classification demandée (reconnaissance des présidents français). Cela ne veut pas dire pour autant que la cooccurrence ne sera pas utile pour classer les textes (les autres couches comme la convolution peuvent réintroduire cette notion plus tard dans le réseau, voir Section 3), mais cela indique que pour la représentation des mots et la discrimination des locuteurs, le

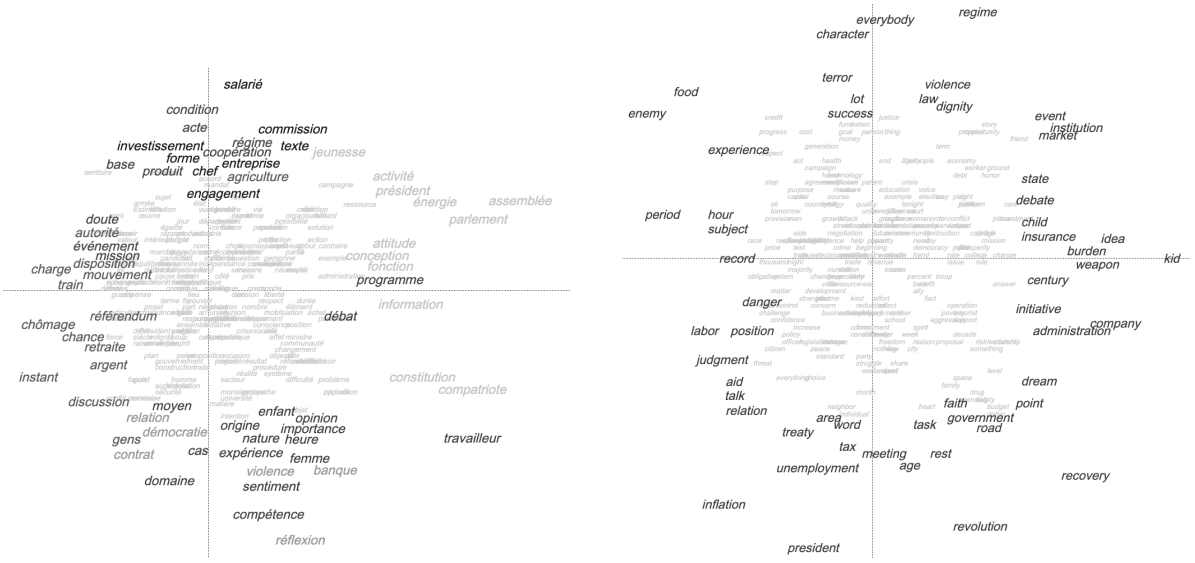


FIGURE 2.10 – AFC du tableau des mots décrits par leurs coordonnées issues de l'embedding (*Word2Vec*) ajusté après apprentissage sur le corpus des présidents français

deep learning préfère un autre type d'espace vectoriel. Et pour comprendre et extraire de l'information de cette nouvelle représentation des mots, il est nécessaire de faire quelques calculs sur la distance euclidienne des mots dans ce nouvel espace.

Rappelons que *Word2Vec* propose une représentation qui reflète la proximité sémantique (figure 2.3). Cette méthode intuitive a d'ailleurs été largement plébiscitée ces dernières années notamment car elle permet de repérer dans le texte des analogies sémantiques avec de simples calculs vectoriels du type : $roi - homme + femme = reine$ (figure 2.3) ; ou encore des calculs de cooccurrences spécifiques avec des approximations comme les k *plus proches voisins* (*k-nearest neighbors* (KNN) en anglais). C'est d'ailleurs cette notion de plus proches voisins qu'il est proposé d'utiliser ici pour donner du sens à notre *embedding* final. Prenons l'exemple du mot *territoires*, un des mots les plus spécifiques du discours d'E. Macron (figure 2.11). Le calcul des k plus proches voisins sur l'*embedding* de *Word2Vec* donne les mots : *ruraux*, *outre-mer*, *métropole*, *territoriale*, *départements*, *régions*. Cette liste est très proche de celle donnée par le calcul statistique des cooccurrences spécifiques (figure 2.11) ; les deux méthodes convergent.

Si on projette maintenant cette liste de mots dans une distribution statistique entre les différentes parties du corpus (les présidents français), on observe qu'il n'y pas vraiment de corrélation entre le partitionnement du corpus et la proximité sémantique de ces mots (figure 2.12). Ce constat est logique puisque le calcul de cooccurrences spécifiques ou *Word2Vec* ne prend pas en compte le partitionnement.

Si on extrait maintenant les représentations vectorielles (l'*embedding* des mots) après la correction des poids du réseau (l'entrainement), le résultat du calcul des k plus proches

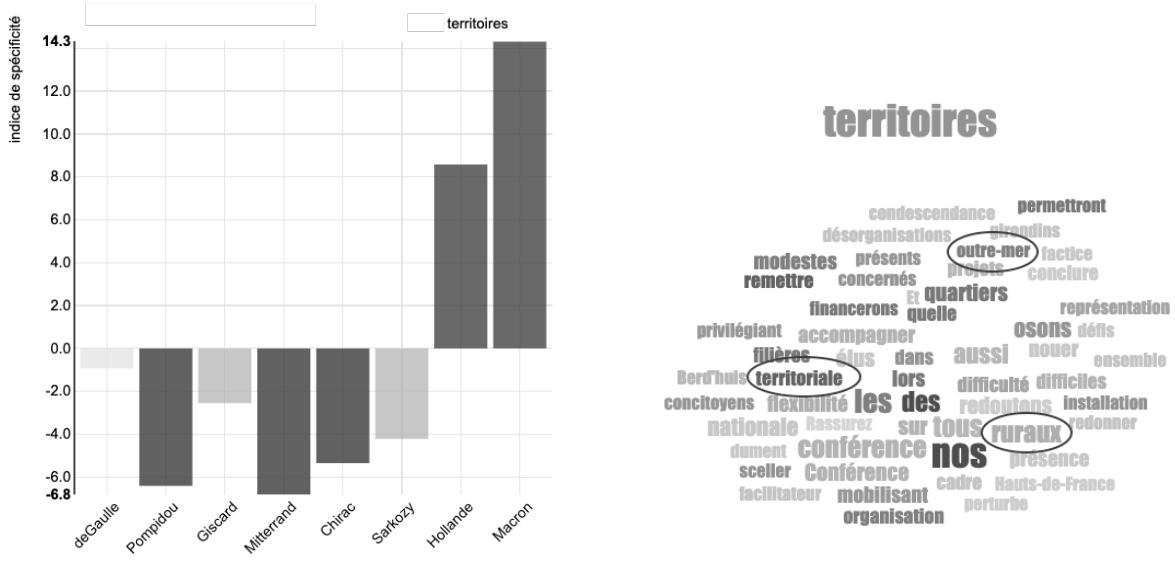


FIGURE 2.11 – Distribution du mots *territoires* et cooccurences spécifiques chez E. Macron.

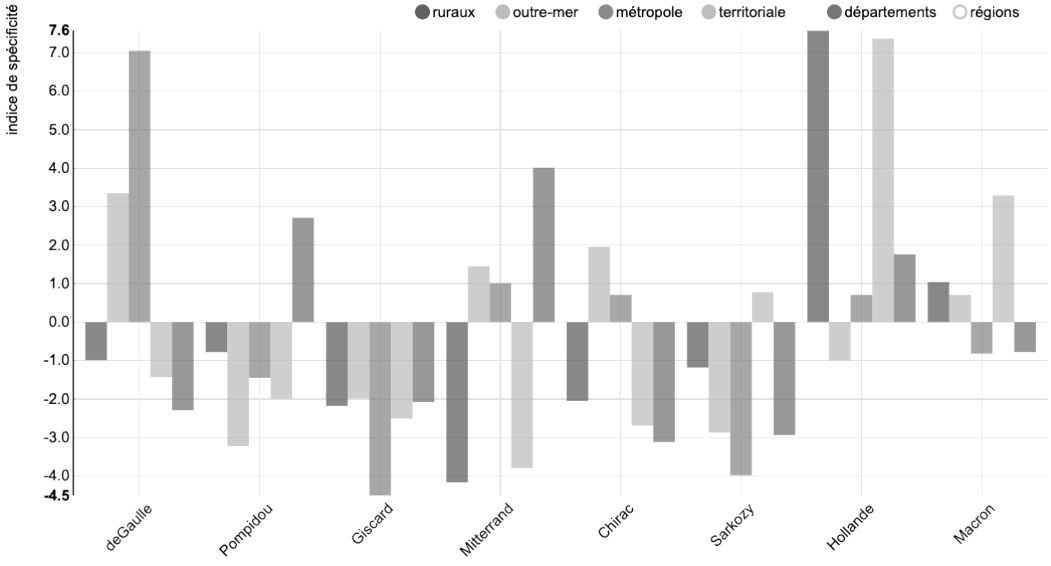


FIGURE 2.12 – k plus proches voisins du mot *territoires* - *Word2Vec*

voisins est tout à fait différent. Pour le même mot *territoires* on obtient : *sécuritaire*, *réorientation*, *lâches*, *finance*, *réorienter*. Cette liste n'a plus beaucoup de sens du point de vue de la cooccurrence. En revanche, si on projette cette nouvelle liste dans la distribution statistique du corpus, cette liste devient tout à fait pertinente d'un point de vue contrastif (figure 2.13).

Il semble ici que le réseau ait regroupé les mots par spécificité statisitque (un *z-score* par

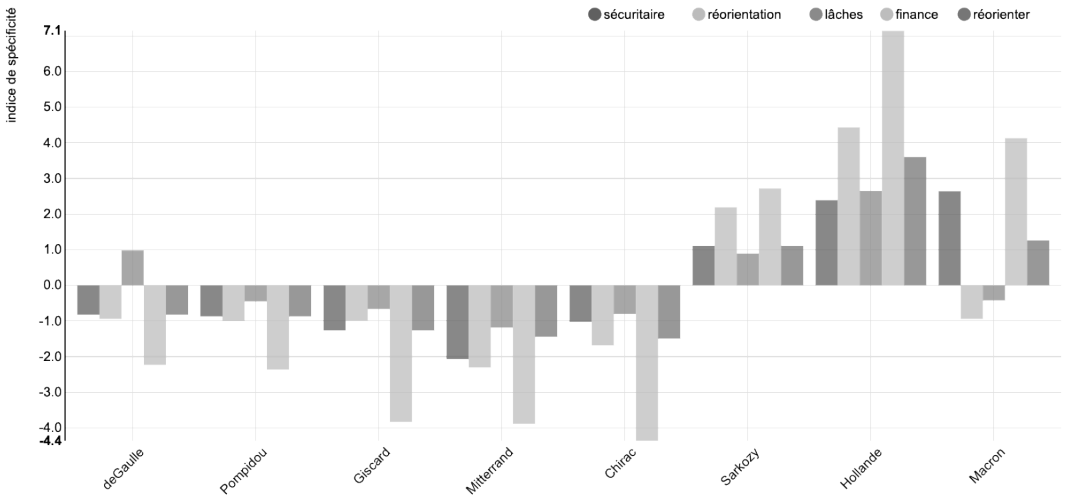


FIGURE 2.13 – k plus proches voisins du mot *territoires* - après classification

exemple). Chaque classe, chaque président, occupe une zone dans l'espace vectoriel. Cette nouvelle organisation est celle retenue par le réseau pour pouvoir classer les textes avec une précision correcte.

L'*embedding*, une fois corrigé, est donc un moyen efficace d'entrevoir les choix du modèle qui ont conduit à produire une classification. Dans la communauté du deep learning, cet espace vectoriel est surtout un moyen d'optimiser le temps de calcul et d'améliorer les scores d'*accuracy*. Chaque année, de nouveaux modèles sont proposés qui repoussent les limites de ce préapprentissage. Mais quel que soit le modèle choisi, l'observation de l'*embedding* doit se faire en fin d'apprentissage (puisque les poids de cet *embedding* sont modifiables par le modèle). La compréhension de cet espace vectoriel créé pour la classification est essentielle pour la prise en main du reste de notre architecture. Dans nos prochaines analyses, chaque mot mis en évidence doit être considéré comme une zone de cet espace vectoriel (le mot plus ses plus proches voisins) et non un token unique.

Pour conclure à ce stade, nous avons une idée de l'organisation des mots dans notre modèle, nous allons voir maintenant comment le réseau utilise cette information pour se construire une véritable représentation abstraite du texte et comment extraire de cette nouvelle forme du texte des marqueurs linguistiques interprétables pour l'humain.

Chapitre 3

Convolution et déconvolution

Ce chapitre reprend en partie la section 3 du chapitre d'ouvrage Vanni *et* Precioso *2021, complété par une traduction de l'article* Vanni, Ducoffe *et al. 2018 disponible en annexe.*

3.1. Abstraction des données

L'idée de proposer un *deep learning* descriptif vient de l'hypothèse que les performances de l'IA en classification de texte sont liées à la détection d'unités linguistiques plus riches que les seuls tokens et sur lesquelles les calculs statistiques standards n'ont pas la main. Les phénomènes complexes autres que la fréquence des mots proposés par Mellet et D. Longrée 2009 suggèrent que nous puissions un jour découvrir de nouveaux observables linguistiques. Le *deep learning* est, de par sa nature, un excellent candidat pour observer ces objets linguistiques profonds jusqu'ici seulement pressentis.

Les réseaux neuronaux convolutifs choisis ici considèrent le texte sous une forme statique (une image), tout en introduisant une notion séquentielle avec une fenêtre glissante qui analyse les mots dans leur contexte. C'est cette approche particulière contextualisante qui diffère des calculs essentiellement fréquentiels de l'ADT. Le pari est que la convolution, pour atteindre un tel niveau de performance, doit être capable d'analyser des réalités linguistiques complexes sur la chaîne : segments phraséologiques, motifs discontinus, multi-cooccurences.

Revenons sur le fonctionnement de la convolution. Nous avons vu que ce type de réseau de neurones utilise un mécanisme de filtres qui se spécialisent et repèrent les marqueurs spatiaux qui permettent de discriminer les données. Dans le cas d'une classification d'images, il est commun de voir apparaître dans ces filtres les contours d'une image ou encore des formes plus représentatives comme le nez, les yeux ou la bouche dans le cas d'une reconnaissance de visages ou les oreilles, la queue, les moustaches dans le cas d'une classification d'espèces animales (figure 3.1). On parle alors d'abstraction des données. Le réseau est capable de se créer une représentation abstraite dans le but de discriminer efficacement l'information.

Globalement, la convolution sur le texte fonctionne de la même manière. Des filtres sont

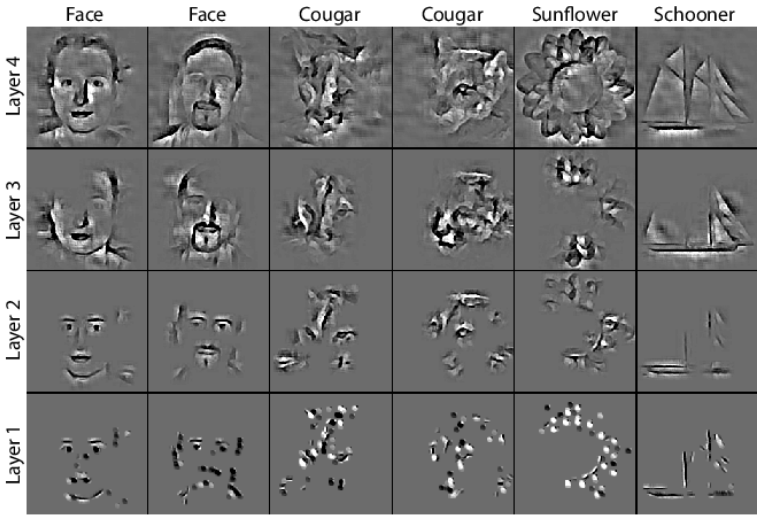


FIGURE 3.1 – Classification d'image par convolution et abstraction des données. Source : `https://api.semanticscholar.org/CorpusID:975170`

appliqués et capturent l'information nécessaire et suffisante à la classification. La seule différence concerne la dimension des données. L'image est un type donné à deux dimensions (hauteur, largeur), alors que le texte n'a qu'une seule dimension qui représente la séquence de mots. Cette différence modifie légèrement le mécanisme convolutionnel. Les filtres ne se déplacent plus de gauche à droite et de haut en bas comme sur une image, mais seulement de mot en mot en parcourant la séquence. Même si l'embedding introduit une dimension supplémentaire (chaque mot est représenté par un vecteur), il n'entre pas en jeu dans le glissement des filtres de la convolution qui analyse chaque mot avec son profil vectoriel complet.

De même que pour les images, le réseau construit avec la convolution une représentation abstraite du texte. Une représentation qui prend évidemment en compte le poids fréquentiel des mots comme en statistique, mais aussi le contexte ou la séquentialité du texte lorsque la fréquence des mots n'est plus suffisante pour classer un texte. Un des objectifs majeurs du *deep learning* en ADT est donc de pouvoir décrypter ces filtres et retranscrire cette information générée par le réseau sous une forme interprétable pour l'humain.

Pour y parvenir, il faut introduire une notion supplémentaire à notre modèle, la *déconvolution*. Cette méthode agit comme un décodeur pour les réseaux à convolution. Adaptée au texte cette *déconvolution* propose une nouvelle manière de lire les textes sous une forme abstraite proposée par le *deep learning*.

3.2. Text Deconvolution Saliency (TDS)

La question soulevée ici et à laquelle nous entendons répondre est de savoir si des modèles linguistiques complexes sont effectivement exploités implicitement dans les architectures profondes pour conduire à de meilleures performances. Les réseaux neuronaux utilisent-ils les cooccurrences et d'autres méthodes standards considérées dans l'analyse traditionnelle

des données textuelles (ADT) ? S'appuient-ils également sur une structure linguistique complémentaire, invisible pour les techniques statistiques classiques ? Si tel est le cas, la projection des caractéristiques d'un de ces modèles sur le texte mettrait en évidence de nouvelles structures linguistiques et permettrait alors d'améliorer l'analyse de corpus et de mieux comprendre la puissance des techniques d'apprentissage profond.

L'hypothèse de départ est que le *deep learning* est sensible aux unités linguistiques sur lesquelles reposent des calculs statistique comme le *z-score* par exemple, mais aussi à des phénomènes autres que la simple fréquence des mots, donnant lieu à des observables linguistiques plus complexes. Pour la vérifier, nous allons confronter ici l'analyse des données textuelles classique (statistique) et les réseaux neuronaux convolutifs.

La méthode choisie utilise les réseaux de *déconvolution* issus de l'analyse d'images. L'idée est d'en tirer une nouvelle perspective pour l'analyse de texte que nous appelons **Text Deconvolution Saliency (TDS)**. Cette saillance textuelle permise par déconvolution correspond à la somme des activations des neurones qui encodent chaque mot en sortie d'une convolution. Un tel score fournit une représentation des mots qui met en évidence des motifs pertinents utilisés par le modèle pour la décision de classification. Le *z-score* (voir section 3.2.4) et le **TDS** sont ainsi comparés en utilisant trois langues différentes : anglais, français et latin. Nous allons voir que pour tous ces jeux de données, le **TDS** met en évidence de nouveaux observables linguistiques, invisibles avec le *z-score* seul.

3.2.1. Contexte

Les réseaux neuronaux convolutifs (CNN) sont largement utilisés dans la communauté de l'analyse d'images assistée par ordinateur pour un large panel de tâches, allant de la classification ou la détection d'objets jusqu'à leur représentation sémantique. Il s'agit d'une approche ascendante où l'on applique à une image en entrée, des couches empilées de convolutions non linéaires et des techniques de sous-échantillonnage.

Encouragés par le succès rencontré dans ces disciplines, les chercheurs ont appliqué les CNN à des problèmes liés aux textes (Nal Kalchbrenner, Edward Grefenstette et Phil Blunsom 2014 ; Kim 2014b). L'utilisation des CNN pour la modélisation de phrases remonte à (R. Collobert et Weston 2008). Collobert a adapté les CNN à divers problèmes de NLP, notamment la *tokenization* (découpage des mots), l'analyse morpho-syntaxique, la lemmatisation, la reconnaissance d'entités nommées ou encore l'étiquetage sémantique. Les CNNs pour le traitement du texte fonctionnent comme une analogie entre une image et une représentation textuelle. En effet, chaque mot est intégré dans une représentation vectorielle, puis plusieurs mots (un texte) forment une matrice (concaténation des vecteurs) semblable à une image formée de pixels.

D'un autre côté, les réseaux de neurones récurrents (principalement GRU et LSTM) sont eux aussi connus pour leurs bonnes performances dans un large éventail de tâches pour le texte. Mais des comparaisons récentes ont confirmé l'avantage des CNNs par rapport aux RNN lorsque la tâche à accomplir est essentiellement une tâche de classification (Yin et al. 2017).

En analyse de texte, on cherche en général à mettre en évidence des schémas linguistiques qui font apparaitre les contrastes d'un corpus : spécificités et similitudes (Feldman, R., and Sanger, J. 2007 ; Lebart, L. and Salem, A. and Berry, L. 1998). Ce type d'analyse s'appuie principalement sur des méthodes basées sur les fréquences des mots telles que la mesure du *z-score*. Cependant, ces méthodes ont jusqu'à présent échoué à mettre en évidence des marqueurs linguistiques plus complexes. Ces marqueurs n'ont d'ailleurs pour certains pas encore été observés empiriquement, comme par exemple les *motifs syntaxiques* (Mellet et D. Longrée 2009).

Dans ce contexte, la classification supervisée à partir de CNNs, peut être exploitée pour l'analyse de corpus. En effet nous l'avons vu, les CNNs apprennent automatiquement des critères du texte pour regrouper les unités similaires et écarter les unités qui se rapportent à des classes différentes. Finalement, la prédiction (classification) d'un modèle repose sur des caractéristiques qui infèrent les spécificités et les similarités dans un corpus. La projection de ces caractéristiques dans le texte révèle des passages pertinents et peut automatiser la découverte de nouvelles structures linguistiques comme les *motifs syntaxiques* cités précédemment. En outre, les CNNs présentent d'autres avantages pour l'analyse linguistique. Ce sont des architectures statiques qui sont plus robustes au problème de la disparition du gradient (*vanishing gradient problem*), et peuvent donc également modéliser la dépendance à long terme dans un texte : (Dauphin et al. 2017 ; Wen et al. 2017 ; Adel et Schütze 2017). Une telle propriété peut aider à détecter les structures linguistiques complexes s'appuyant sur différentes parties du texte. Tous les travaux précédents ont convergé vers une évaluation commune : les CNNs et les RNNs fournissent tous deux des informations pertinentes, mais de nature différente, pour la classification de textes. Cependant, bien que plusieurs travaux aient étudié les structures linguistiques inhérentes aux RNNs, à notre connaissance, aucun d'entre eux ne s'est concentré sur les CNNs.

Concernant l'interprétabilité des modèles, plusieurs axes de recherches ont déjà été étudiés en profondeur. Par exemple la représentation vectorielle des mots (*embedding*) et de leurs poids sémantiques : (Ji et Eisenstein 2014). D'autre part avec les RNNs, Karpathy, Johnson et Fei-Fei 2015 ont démontré l'existence de dépendances entre les mots sur des données textuelles réelles. Enfin Li et al. 2015 ont aussi fourni de nouveaux outils de visualisation pour les modèles récurrents. Ils utilisent des décodeurs de type *t-SNE* afin de mettre en lumière le fonctionnement des modèles neuronaux. La perspective de nos travaux diffère légèrement de ces approches. L'intention principale ici n'est pas d'expliquer comment les modèles fonctionnent, mais plutôt d'utiliser leurs caractéristiques pour fournir des informations complémentaires pour l'analyse linguistique de données textuelles.

Au-delà de l'utilisation très courante des RNNS, il faut remarquer qu'il existe aussi des outils de visualisation pour l'analyse des CNNs, inspirés par le domaine de l'analyse d'image. Les modèles de visualisation (interprétation) de données en classification d'images consistent principalement à afficher les couches cachées afin de repérer les régions actives et pertinentes pour la décision de classification. Pour cela il est possible soit d'entrainer un réseau décodeur, soit utiliser la *backpropagation* sur les données d'entrée pour

mettre en évidence les caractéristiques les plus pertinentes. Bien que ces méthodes puissent contenir des informations précises, elles présentent deux inconvénients principaux : i) elles sont couteuses en calcul : la première méthode nécessite l'apprentissage d'un modèle pour chaque représentation latente, et la seconde repose sur la *backpropagation* pour chaque échantillon de texte soumis. ii) Elles dépendent fortement des hyperparamètres et peuvent nécessiter un ajustement long et couteux en fonction de la tâche à accomplir.

Les réseaux de déconvolution, proposés par Zeiler et Fergus 2014, fournissent une méthode bien plus directe et prête à l'emploi pour projeter les informations détenues par la couche de convolution dans le texte en entrée. Le principe consiste à inverser chaque couche de convolution de manière itérative, pour revenir à l'espace d'entrée, le texte. L'inverse d'une convolution discrète est difficile à calculer. En réponse, une approximation grossière peut être employée qui consiste à inverser le poids des filtres dans une nouvelle couche convolutionnelle, puis à transposer le noyau la matrice. Plus de détails sur le calcul euristique de déconvolution sont fournis dans la section 4.4. La déconvolution présente plusieurs avantages. Tout d'abord, elle induit des exigences minimales en termes de calcul par rapport aux méthodes de visualisation précédentes. De plus, elle a été utilisée avec succès pour la segmentation sémantique d'images (Noh, Hong et Han 2015) qui démontre l'efficacité des réseaux de déconvolution pour identifier les étiquettes de classe pour chaque pixel d'une image.

3.2.2. Modèle

Le modèle proposé est basé sur un réseau convolutif avec une couche d'*Embedding* pour la représentation des mots, une couche de convolution associée à une couche de regroupement (*Pooling*) et deux couches *Dense* (entièrement connectées). La taille de la sortie finale correspond au nombre de classes. Le modèle est entrainé par entropie croisée (ou *cross entropy*) avec l'algorithme de descente du gradient *Adam*.

La figure 3.2 montre la structure globale de notre architecture. L'entrée est une séquence de mots $w_1, w_2 ... w_n$ et la sortie contient les probabilités de chaque classe (pour la classification de texte).

L'*Embedding* est construite à partir d'une architecture *Word2Vec* avec un modèle *Skip-gram*. Cet *Embedding* est également ajusté par le modèle afin d'augmenter la précision.

En analyse d'images, les données sont considérées comme des signaux isotropes à deux dimensions. Une représentation textuelle peut également être considérée comme une matrice : chaque mot est intégré dans un vecteur caractéristique (*Embedding*) et leur concaténation permet de construire une matrice (un texte). Cependant, il ne faut pas supposer que les deux dimensions - la séquence de mots et leurs représentations vectorielles - sont ici isotropes. Ainsi, les filtres des CNNs pour le texte diffèrent de leurs homologues conçus pour les images.

Par conséquent, dans le texte, la largeur du filtre est généralement égale à la dimension de l'*Embedding*, comme l'illustrent les filtres rouge, jaune, bleu et vert de la figure 3.2.

L'utilisation des CNNs présente un autre avantage dans notre contexte : en raison des opérateurs mathématiques de convolution, le calcul peut être facilement parallélisé et par conséquent produit relativement rapidement par un CPU, ce qui est une solution pratique pour éviter l'utilisation de GPU couteux au moment du test.

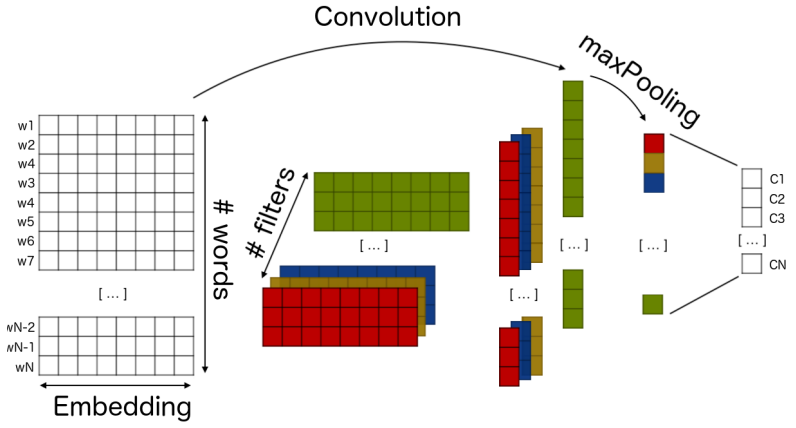


FIGURE 3.2 – CNN appliqué à la classification de textes

3.2.3. Déconvolution

L'application des réseaux de déconvolution au texte n'est pas évidente. Habituellement, en analyse d'image, la déconvolution est représentée par une convolution dont les poids dépendent des filtres du CNNs : ces poids sont inversés et le noyau de la matrice finale est ensuite transposé. Lorsque l'on considère la déconvolution pour le texte, la transposition des matrices n'est pas réaliste puisque nous avons affaire à des dimensions différentes - les séquences de mots et la dimension du filtre. La matrice n'est donc pas transposable.

Un autre inconvénient concerne la dimension de la représentation du texte. Ici cette représentation correspond à la sortie de la convolution avant l'application du *max pooling* (couche de regroupement). Sa dimension est égale au nombre de mots multiplié par le nombre de filtres. La largeur de chaque filtre (rouge, jaune, bleu et vert dans la fig 3.2) correspond à la dimension de l'*embedding*. Pour revenir à un espace similaire à l'*embedding* de départ il faut suréchantillonner (*upsampling*) la sortie de la convolution pour obtenir une nouvelle matrice tridimensionnelle qui correspond au nombre mots, à la taille de l'*embedding* et au nombre de filtres.

Pour analyser la pertinence d'un mot dans un texte, nous ne gardons qu'une seule valeur par mot. Cette valeur correspond à la somme sur l'axe de l'*embedding* de la matrice en sortie de déconvolution. Nous appelons cette méthode le *Text Deconvolution Saliency* (**TDS**). L'algorithme est résumé avec la figure 3.10.

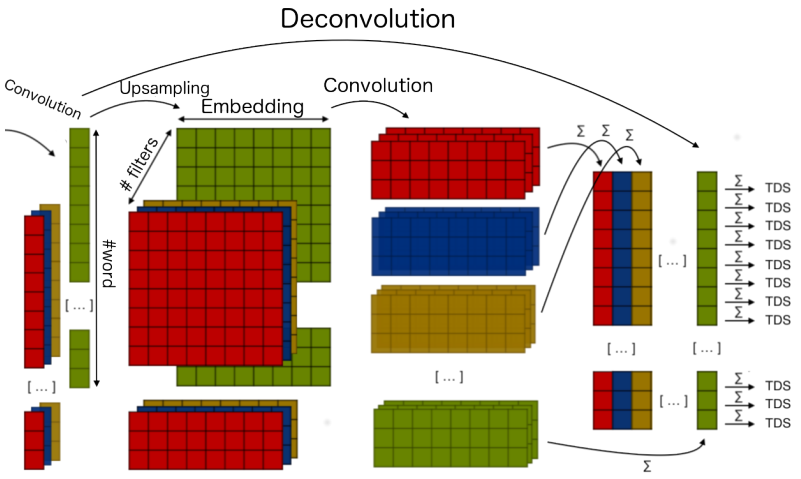


FIGURE 3.3 – Textual Deconvolution Saliency (**TDS**)

Finalement, chaque mot dans un texte a un score **TDS** unique dont la valeur est liée aux autres mots (sur le principe de la fenêtre coulissante de la convolution). Pour analyser la pertinence du **TDS**, nous allons démontrer dans la section suivante de manière empirique que le **TDS** met en évidence des objets linguistiques complexes comme des cooccurrences et peut-être aussi des notions plus fines de syntaxe ou de grammaire.

3.2.4. Experiences

Afin de comprendre la nature des marqueurs linguistiques identifiés par les réseaux de neurones convolutifs, plusieurs tests sont effectués sur différentes langues et le modèle semble obtenir le même comportement dans toutes ces langues. Pour mettre en pratique la partie statistique de ces tests, le logciel Hyperbase Web (voir chapitre 8) est utilisé. Cette plateforme web permet notamment la création de bases de données à partir de corpus textuels, l'analyse et les calculs tels que le *z-score*, les cooccurrences, l'ACP ou encore des méthodes de partitionnement type *K-Means*. Pour évaluer le **TDS** c'est le *z-score* qui a été choisi comme référence pour l'analyse de données textuelles. Les analyses sont contraintes et ne présentent que les cas sur lesquels le *z-score* échoue à classer un texte [1] au profit du **TDS**. À noter qu'aucun texte de l'ensemble de test ne présente le cas inverse, à savoir, correctement classé par le *z-score* et faussement classé par le **TDS**. Les mots en rouge dans les exemples étudiés correspondent aux **TDS** les plus élevés.

Le premier jeu de données utilisé est un corpus bien connu en analyse de sentiments, il s'agit du corpus IMDB composé de 25 000 critiques de films étiquetées par sentiment positif ou négatif, avec environ 230 000 mots.

Le deuxième jeu de données cible les discours politiques français. Il s'agit d'un corpus de 2,5 millions de mots de présidents français qui commence en 1958 (avec C. De Gaulle, premier président de la Cinquième République) et se termine en 2018 avec les premiers

1. ici la classe choisie est celle qui obtient la somme de *z-score* la plus élevés

discours d'E. Macron. Dans ce corpus, nous avons retiré le discours d'E. Macron du 31 décembre 2017, pour l'utiliser comme jeu de données de test. La tâche d'entrainement consiste à reconnaitre chaque président français.

Le dernier jeu de données utilisé est composé des textes latins. Ce corpus contrastif compte 2 millions de mots avec 22 des principaux auteurs du latin classique. Comme pour le jeu de données français, la tâche d'apprentissage consiste à reconnaitre chacun des auteurs en fonction de nouvelles séquences de mots.

Les premiers résultats présentés utilisent le latin avec un extrait du chapitre 26 du 23ème livre de Tite-Live

> *[...] tutus tenebat se quoad multum ac diu obtestanti quattuor milia peditum et quingenti equites in supplementum missi ex Africa sunt . tum refecta tandem spe* ***castra propius hostem*** *mouit classem que et ipse instrui parari que iubet ad insulas maritimam que oram tutandam . in* ***ipso impetu*** *mouendarum de [...]*

Corrélation statistique

Le *z-score* est l'une des métriques standards utilisées en analyse de données textuelles, notamment pour mesurer l'importance des mots ou encore les cooccurrences (C. D. Manning et Schütze 1999). Ce score résulte de la comparaison entre la fréquence observée des mots et la fréquence attendue dans le cas d'une distribution "normale" (loi normale et hypothèse nulle d'equidistribution des mots). Dans le contexte de l'analyse de corpus constatifs, ce calcul appliqué à l'ensemble du vocabulaire peut facilement fournir des informations discriminantes comme une surutilisation de certains mots pour un auteur donné. Il s'agit d'une méthode simple, mais efficace pour analyser les caractéristiques d'un texte. Nous l'utilisons également ici pour classer des textes en fonction d'un *z-score* dit global qui correspond à la somme des scores pour l'ensemble des mots du texte.

Nous pouvons donc utiliser le *z-score* global comme une métrique très simple pour la classification d'auteurs. La paternité d'un extrait donné est ainsi attribuée à l'auteur correspondant au *z-score* global le plus élevé sur cette phrase par rapport à tous les autres *z-score* globaux. La précision moyenne de la prédiction du bon auteur pour chaque phrase est d'environ 87% sur le corpus latin, ce qui confirme que le *z-score* est effectivement une métrique robuste pour l'analyse de contraste dans un corpus. Le CNN quant à lui atteint en général une précision supérieure à 90% pour les mêmes ensembles de tests (voir tableau 3.1).

Cette comparaison confirme que les CNNs sont une alternative très efficace pour reconnaitre certaines des spécificités linguistiques utiles à la discrimination des textes. Des travaux antérieurs concernant le traitement des images ont déjà souligné le rôle clé des couches convolutionnelles dans l'apprentissage de différents niveaux d'abstraction des données pour faciliter leur classification. La question est : quelle est la nature de ces abstractions sur le texte ?

	z-test	Deep Learning
Latin	84%	93%
French	89%	91%
English	90%	97%

TABLE 3.1 – Test d'*accyracy* du *z-score* et du *deep learning*

Comme attendu le **TDS** montre que les CNNs détectent principalement des mots avec un *z-score* élevé. Mais ce type de marqueurs n'est pas la seule structure détectée. Pour rendre les deux approches comparables, il faut normaliser les deux scores (le **TDS** et le *z-score*). Dans le cas du *z-score* les valeurs peuvent être soit positives, soit négatives. On distingue deux seuils [2]. Au-dessus de 2, un mot est considéré comme spécifique et au-dessus de 5, il est fortement spécifique. Pour le **TDS** le score correspond plus à une de force d'activation.

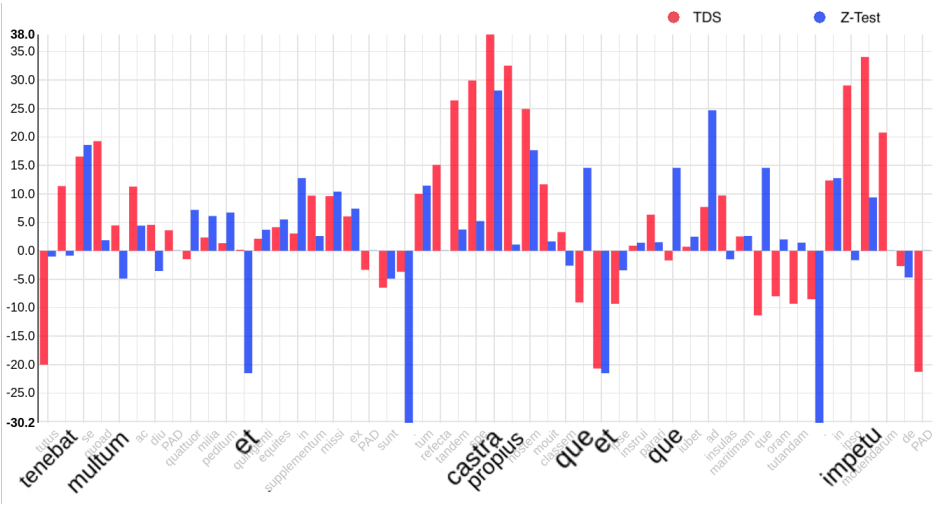


FIGURE 3.4 – *z-score* versus **TDS** - Exemple : Tite-live Livre XXIII Chap. 26

La figure 3.4 nous montre une comparaison entre le *z-score* et le **TDS** sur l'exemple du corpus latin (Tite-live - Livre XXIII Chap. 26 cité précédemment). Cet extrait illustre très bien la notion de passage-clé spécifique dans le cas de Tite-Live [3]. À première vue lorsque le *z-score* est élevé, le **TDS** l'est aussi. On note aussi que le **TDS** est également élevé pour les mots voisins (par exemple autour du mot *castra*). Cependant, ce n'est pas toujours le cas : par exemple, de petits mots comme *que* ou *et* ont un *z-score* élevé mais ils ne produisent pas le même impact sur **TDS**. Des mots comme *tenebat*, *multum* ou *propius*

2. Le *z-score* converge vers une loi normale lorsque les données sont suffisamment grande. Le résultat que nous obtenons par le *z-score* est un écart-type. Un faible écart-type indique que les points de données ont tendance à être proches de la moyenne (la valeur attendue). Au-delà de 2, ce score signifie qu'il y a moins de 2% de chance d'avoir cette distribution. Au-dessus de 5, c'est moins de 0,1%

3. Tite-Live est l'un des plus célèbres historiens connus pour son oeuvre monumentale de l'histoire de Rome depuis sa fondation (142 livres)

ont même des poids opposés (figure 3.4). Le coefficient de corrélation de Pearson [4] indique que dans cet exemple, il n'y a pas de corrélation linéaire entre le *z-score* et le **TDS** (le coefficient de Pearson atteint seulement 0,38). Pourtant cet extrait est l'un des exemples les plus corrélés de notre ensemble de données. Il semble donc que le CNN présente un niveau d'abstraction du texte différent du *z-score*.

Unités textuelles complexes

Le corpus de critiques de films IMDB (corpus anglais de classification des sentiments) donne des premiers éléments de réponses concernant les marqueurs linguistiques utilisés par le CNN. Avec les méthodes statistiques standards, nous pouvons facilement mettre en évidence le vocabulaire spécifique de chaque classe (positif/négatif). En utilisant le *z-score*, on trouve par exemple que les mots *too*, *bad*, *no* ou *boring* sont surreprésentés dans la classe des sentiments négatifs, et inversement les mots *and*, *performance*, *powerful* ou *best* le sont pour la classe des sentiments positifs. Est-ce suffisant pour détecter automatiquement si un nouvel avis est positif ou non ? Voyons un exemple avec un extrait d'une critique de décembre 2017 (issus du corpus de test) sur un des derniers blockbusters américains :

> *[...]* ***i enjoyed three moments*** *in the film in total ,* ***and if i am being honest and*** *the person* ***next to me fell asleep*** *in the middle and started snoring during the slow space chasescenes .* ***the story failed to*** *draw me in and entertain* ***me the way*** *[...]*

En général, le *z-score* est suffisant pour prédire la classe de ce type de commentaire. Mais dans cet exemple, le CNN semble faire mieux, mais pourquoi ?

Si l'on additionne tous les *z-score* (positifs et négatifs), la classe positive obtient un score plus important que la négative. Les mots *movie*, *and*, *honest* et *entertain* (avec des *z-scores* respectivement de 5,38, 12,23, 4 et 2,4) rendent cet exemple positif. Cependant ici le CNN active différentes parties dans l'extrait (mots en rouge dans l'exemple). Ainsi dans la sous-séquence *and if I am being honest and*, il y a deux occurrences de *and* mais la première est suivie de *if* et le z-score pour cette expression tombe à -0,84 pour *and if* sur la classe positive ce qui est loin des 12,23 pour le mot *if* seul. Une autre analyse donne quelques informations complémentaires concernant la prise de décision du réseau. L'analyse de la poly-cooccurrence [5] (Vanni et Mittmann 2016) autour de l'expression *and if* (figure 3.5), met en évidence le mot *honest* comme l'un des adjectifs les plus spécifiques associés à *and if* comme dans cet exemple.

4. Le coefficient de corrélation de Pearson mesure la relation linéaire entre deux ensembles de données. Sa valeur est comprise entre +1 et −1, où 1 correspond à une corrélation linéaire positive totale, 0 à une absence de corrélation linéaire et −1 à une corrélation négative totale

5. Cette analyse montre les principales cooccurrences pour un mot ou une expression donné. Il y a deux couches de cooccurrences, la première (en haut) montre la cooccurrence directe et la seconde (en bas) montre un second niveau de cooccurrence. Ce niveau est donné par le contexte des deux mots (pris ensemble). Les couleurs et les lignes en pointillés ne sont utilisées que pour faciliter la lecture (les lignes en pointillés sont utilisées pour le premier niveau). La largeur de chaque ligne est liée au score du *z-score* (plus le *z-score* est grand, plus la ligne est large).

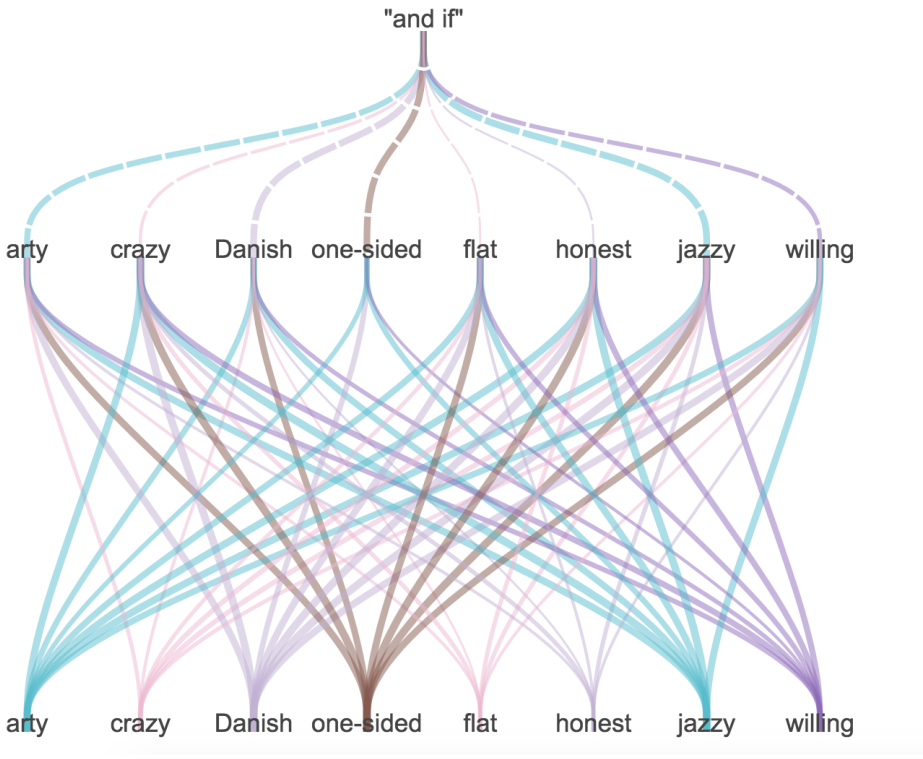


FIGURE 3.5 – Analyse des cooccurrences de l'expression *and if*

De plus, nous avons le même comportement avec le verbe *fail* qui accueille le mot *asleep* juste à côté. *Asleep* seul n'est pas vraiment spécifique de la critique négative (*z-score* de 1,13). Mais l'association des deux mots devient hautement spécifique des phrases négatives (voir l'analyse des cooccurrences - figure 3.6).

Le **TDS** confirme donc ici que le CNN semble se concentrer non seulement sur les *z-score* élevés, mais aussi sur des phénomènes plus complexes. Nous allons voir maintenant que ces observations peuvent être généralisées en utilisant les autres langues de notre ensemble de données.

Lexique, syntaxe et phraséologie

Dans le corpus français, les marqueurs affichés avec le **TDS** se comportent visiblement de la même façon que pour l'anglais. Rappelons que dans ce corpus le discours d'E. Macron du 31 décembre 2017 a été retiré, pour l'utiliser comme jeu de données de test. Dans ce discours, le CNN reconnait naturellement l'actuel président (prediction) et semble réussir à identifier des structures plus complexes et spécifiques d'E. Macron. Par exemple, dans cet extrait :

> *[...] notre pays* ***advienne à*** *l'école pour nos enfants, au travail pour l' ensemble de* ***nos concitoyens*** *pour le climat pour le quotidien de chacune et chacun d' entre vous .* ***Ces transformations profondes*** *ont commencé et se* ***poursui-***

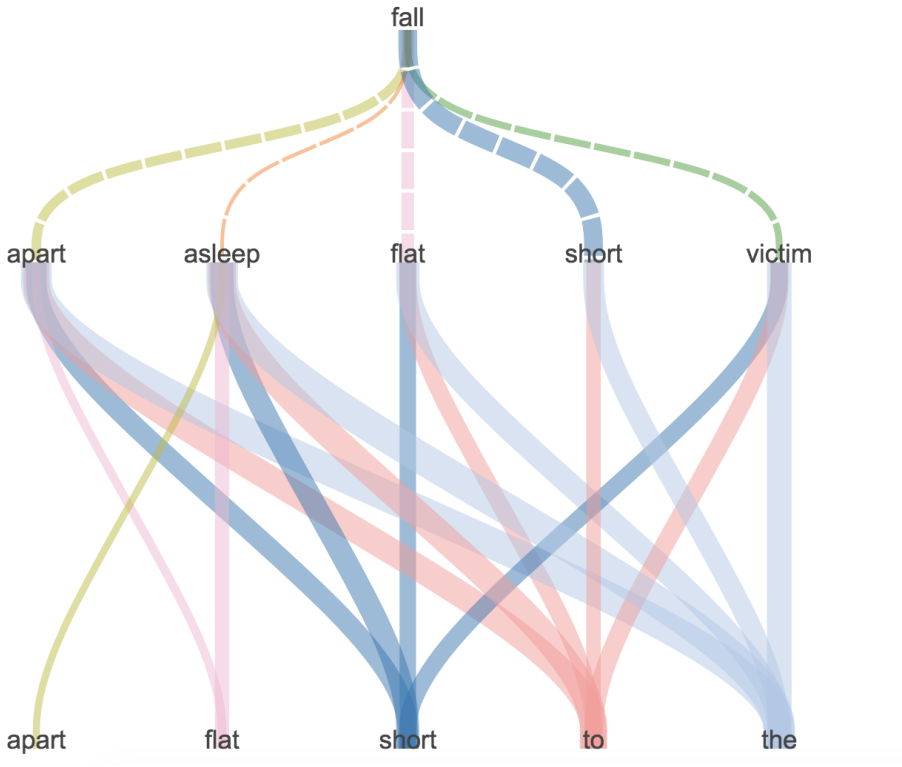


FIGURE 3.6 – Analyse des cooccurrences du mot *fall*

vront *avec la même force le même rythme la même intensité [...]*

Dans cet exemple le *z-score* global est plus élevé pour C. De Gaulle que pour E. Macron. L'erreur d'attribution statistique s'explique par une phraséologie gaulliste et la multiplication de marqueurs linguistiques hautement spécifique de C. De Gaulle : ce président avait la particularité de faire des phrases longues et littéraires articulées autour de conjonctions de coordination comme *et* (*z-score* de 28 pour De Gaulle, deux occurrences dans l'extrait). Son discours est également plus conceptuel que la moyenne, ce qui se traduit par une surutilisation des articles définis *le*, *la*, *l´*, *les* très nombreux dans l'extrait (7 occurrences) ; notamment au féminin singulier (*la république*, *la liberté*, *la nation*, *la guerre*, etc), ici nous avons *la même force*, *la même intensité*.

Les meilleurs résultats obtenu par le CNN peuvent être surprenants pour un linguiste mais correspondent pourtant parfaitement à ce que l'on sait de l'identité linguistique des discours de E. Macron.

La partie de l'extrait qui a le plus d'impact sur le classement CNN (**TDS** élevé) est liée au syntagme nominal *transformations profondes*. Pris séparément, aucun des deux mots de la phrase n'est très macronien d'un point de vue statistique (*z-score transformations* = 1,9 ;*profondes* = 2,9). Mieux, le syntagme lui-même n'apparait pas dans le corpus d'apprentissage du Président (0 occurrence). Cependant, on constate que le *z-score* de la

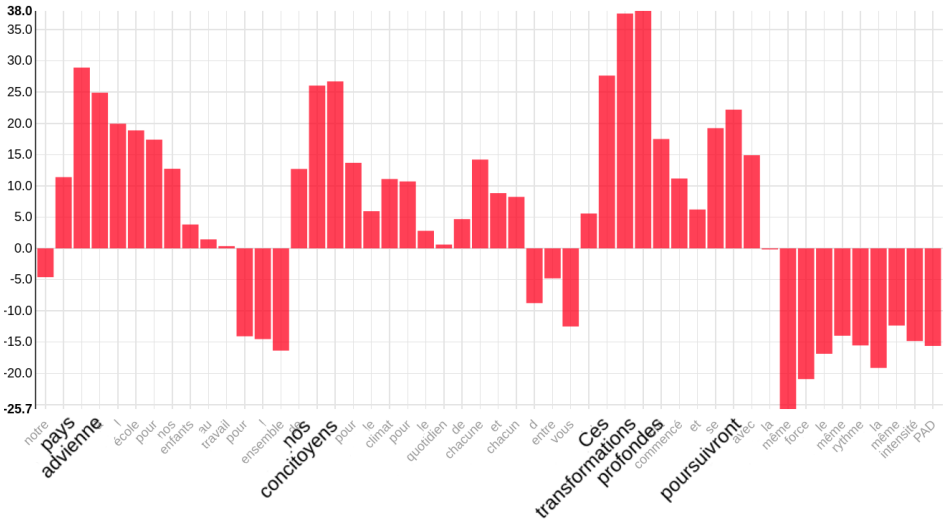


FIGURE 3.7 – Deconvolution appliqué au discours d'E. Macron.

cooccurrence de *transformation* et *profondes* s'élève à 4,81 chez Macron : ce n'est donc pas l'occurrence d'un seul mot, ou de l'autre, qui est macronienne, mais l'apparition simultanée des deux dans la même fenêtre. La deuxième partie de l'extrait met en évidence un **TDS** élevé sur deux verbes *advienne* et *poursuivront*. D'un point de vue sémantique, ces deux verbes contribuent parfaitement (après le passage *transformations profondes*) à donner une dynamique nécessaire à un discours qui prône le changement. Toutefois, ce sont les temps verbaux (portés par la morphologie des verbes) qui apparaissent comme le facteur déterminant dans l'analyse. Le calcul des codes grammaticaux cooccurrents au mot *transformations* indique ainsi que les verbes au subjonctif et les verbes au futur (ainsi que les noms) sont les codes privilégiés de E. Macron (figure 3.8).

Plus précisément, l'algorithme indique que, pour E. Macron, lorsque *transformation* est associé à un verbe au subjonctif (ici *advienne*), alors il y a généralement aussi un verbe au futur dans l'environement proche de celui-ci (ici *poursuivront*). *transformations profondes*, *advienne* au subjonctif, *poursuivront* au futur : tous ces éléments réunis forment un discours prometteur d'action, dans la bouche d'un Président jeune et dynamique. Enfin, le graphique indique que *transformations* est particulièrement associé aux substantifs dans les discours du Président : dans une concentration extraordinaire, l'extrait en recense 11 (*pays*, *école*, *enfants*, *travail*, *concitoyens*, *climat*, *quotidien*, *transformations*, *force*, *rythme*, *intensité*).

Nouveaux observables

Comme pour le jeu de données français, le dernier exemple sur le corpus latin repose sur une tâche de prédicton d'auteurs à partir d'un apprentissage sur un corpus contrastif de 2 millions de mots avec 22 auteurs principaux écrivant en latin classique.

La statistique (*z-score*) identifie l'extrait latin (celui présenté dans la section 4.7) comme

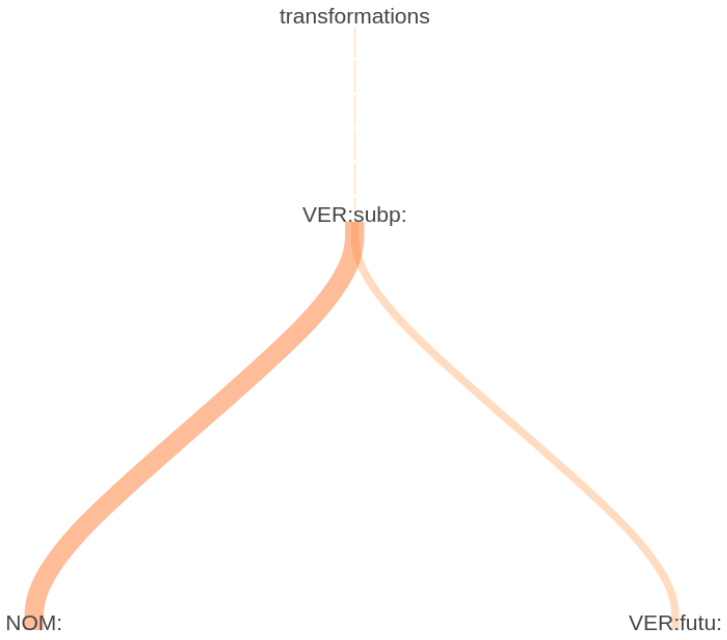


FIGURE 3.8 – Principales catégories grammatical co-occurrentes de *transformations*

appartenant à César [6] . Tite-Live n'est pas loin, en tant qu'historiens, César et Tite-Live partagent un certain nombre de mots spécifiques : par exemple des mots outils comme *se* (pronom réfléchi) ou *que* (un coordinateur) et des prépositions comme *in*, *ad*, *ex*, *of*. Il y a aussi des noms comme *equites* (cavalerie) ou *castra* (camp fortifié).

L'attribution de la phrase à César ne peut pas se baser uniquement sur le *z-score* : *que* ou *in* ou *castra*, avec leurs scores équivalents ou inférieurs à ceux de Tite-Live. En revanche, les écarts de *se*, *ex*, sont plus importants, de même que ceux de *equites*. Deux termes très césariens font sans doute la différence : *iubet* (il ordonne) et *milia* (milliers).

Le *z-score* élevé de *quattuor* (quatre), *castra* (camp), *hostem* (l'ennemi), *impetu* (l'assaut) chez Tite-Live ne suffit pas à faire basculer l'attribution à cet auteur.

En revanche, le CNN active plusieurs zones apparaissant en début de phrase et correspondant à des structures syntaxiques cohérentes (pour Tite-Live) comme *Tandem reflexes spe castra propius hostem mouit* (alors, l'espoir étant enfin revenu, il rapprocha le camp de celui de l'ennemi) – malgré le fait que *castra* dans *hostem mouit* ne soit attesté que par Tacite [7].

On trouve également *in ipso metu* (dans la peur elle-même), tandis que *in* suivi de *metu*

6. Gaius Julius Caesar, 100 av. J.-C. - 44 av. J.-C., généralement appelé Jules César, était un homme politique et un général romain, ainsi qu'un auteur notable de prose latine.
7. Publius (ou Gaius) Cornelius Tacitus, 56 av. J.-C. - 120 av. J.-C., était un sénateur et un historien de l'Empire romain.

n'est compté qu'une fois chez César et une fois aussi chez Quinte-Curce [8].

Des structures plus complexes sont peut-être aussi détectées par le CNN : la structure *tum* + participe Ablatif Absolu (*tum refecta*) est plus caractéristique de Tite-Live (*z-score* de 3,3 avec 8 occurrences) que de César (*z-score* de 1,7 avec 3 occurrences), même si elle est encore plus spécifique de Tacite (*z-score* de 4,2 avec 10 occurrences).

Enfin, et plus probablement, la cooccurrence entre *castra*, *hostem* et *impetu* peut avoir joué un rôle majeur : figure 3.9

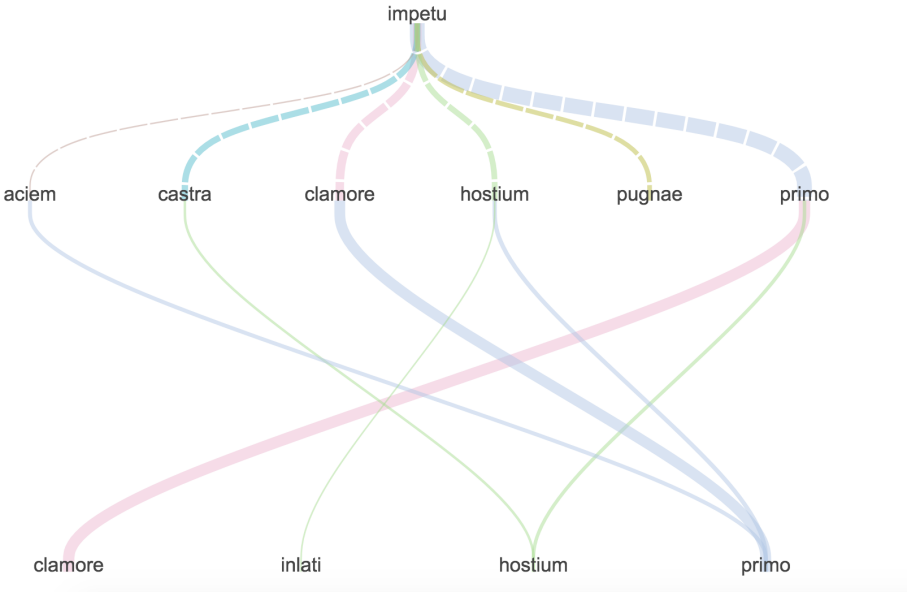


FIGURE 3.9 – Co-occurrences spécifique entre *impetu* et *castra*

Chez Tite-Live, *impetu* apparait comme un cooccurrent avec des lemmes *hostis* (*z-score* de 9,42) et *castra* (*z-score* de 6,75), alors que *hostis* n'a qu'un écart de 3,41 chez César et que *castra* n'apparait pas dans la liste des cooccurrents.

Pour *castra*, le premier cooccurrent de Tite-Live est *hostis*. (*z-score* de 22.72), devant *castra* (*z-score* de 10,18), *ad* (*z-score* de 10.85), *in* (*z-score* de 8.21), *impetus* (*z-score* de 7,35), *que* (*z-score* de 5,86) alors que dans César, *impetus* n'apparaît pas et les scores de tous les autres lemmes sont plus faibles sauf *castra* (*z-score* de 15,15), *hostis* (*z-score* de 8), *ad* (*z-score* de 10,35), *in* (*z-score* de 5,17), *que* (*z-score* de 4.79).
Ainsi, ces derniers résultats suggèrent non seulement que les CNNs parviennent à rendre compte de la spécificité des mots mais aussi d'une structure plus profonde des textes.

8. Quintus Curtius Rufus était un historien romain, probablement du 1er siècle, son seul ouvrage connu et unique survivant étant "Histoires d'Alexandre le Grand"

3.2.5. Conclusion

Le **TDS** est efficace sur un large éventail de corpus. En croisant des approches statistiques avec des réseaux de neurones, une nouvelle stratégie est proposée pour détecter automatiquement des observables linguistiques complexes, jusqu'ici difficilement détectables par des méthodes basées uniquement sur les fréquences. Rappelons que la matière linguistique extraite par le **TDS** ne peut pas être due au hasard : les zones d'activation permettent au modèle d'obtenir des taux de reconnaissance de plus de 91% sur le discours politique français et 93% sur le corpus latin ; deux taux équivalents ou supérieurs aux taux obtenus par le calcul statistique des passages clés.

3.3. Variantes du TDS

Nous l'avons vu, la déconvolution est un ensemble de processus algorithmiques qui visent à inverser les effets de la convolution. La convolution génère une représentation abstraite du texte, l'idée de la déconvolution appliquée au texte est donc de revenir à une représentation concrète du même texte, tout en conservant l'information apprise pendant la phase d'abstraction. Pour rappel, le texte est une matrice dont le nombre de lignes correspond au nombre de mots, et le nombre de colonnes à l'*embedding* de ces mots. L'opération *convolution/déconvolution* peut donc être considérée comme un auto-encoder à l'image de *Word2Vec*, avec en entrée et en sortie une matrice *mot* x *embedding*. Dans cette idée nous allons voir maintenant que la mise en pratique du **TDS** autorise quelques variations d'implémentation. La première est celle présentée précédemment sur base de convolution transposée. Une deuxième solution consiste à introduire une contrainte dans l'architecture du réseau pour pouvoir lire directement les informations dans les couches cachées de la convolution.

3.3.1. Méthode 1 : convolution transposée

Pour résumer, la première méthode repose sur des implémentations de déconvolution que l'on appelle convolution transposée. Proposée à l'origine pour l'analyse d'image, une convolution transposée regroupe deux couches successives. Une première couche d'*unpooling* suivie d'une autre de convolution.

L'*unpooling* a pour rôle d'augmenter la résolution de l'image fabriquée à partir d'une convolution. En effet la convolution repose sur de nombreux hyperparamètres qui affectent la taille des images en sortie, comme la taille des filtres, le *padding* et le *stride* (voir section 3.3.2). à noter qu'une couche de "regroupement" ou *pooling* est généralement appliquée après la convolution pour compresser davantage l'information [9]. Cette compression est une optimisation qui vise à augmenter les performances du modèle. Même si pour les images il n'est pas forcément nécessaire d'utiliser une déconvolution pour interpréter l'abstraction des données (une image même compressée peut être visualisée et analysée, voir figure 3.1), pour le texte il est essentiel de retrouver la forme séquentielle d'origine pour pouvoir l'interpréter. L'*unpooling* recopie donc les lignes si nécessaire (un simple facteur multiplicatif est appliqué) afin d'obtenir une séquence de mots (un nombre de

9. par exemple le maxpooling a pour effet de ne conserver que les valeurs maximums en sortie de convolution. Cette opération divise alors la taille des données par la taille du pooling choisi.

lignes) semblable à celle en entrée du réseau.

La convolution qui suit lisse les effets de l'*unpooling* (recopie des valeurs) et produit une nouvelle image qui accentue les zones contenant les marqueurs linguistiques identifiés par le réseau. Le nombre de filtres utilisés pour cette convolution correspond à la taille du nouvel embedding généré par la sortie de la deconvolution (figure 3.10).

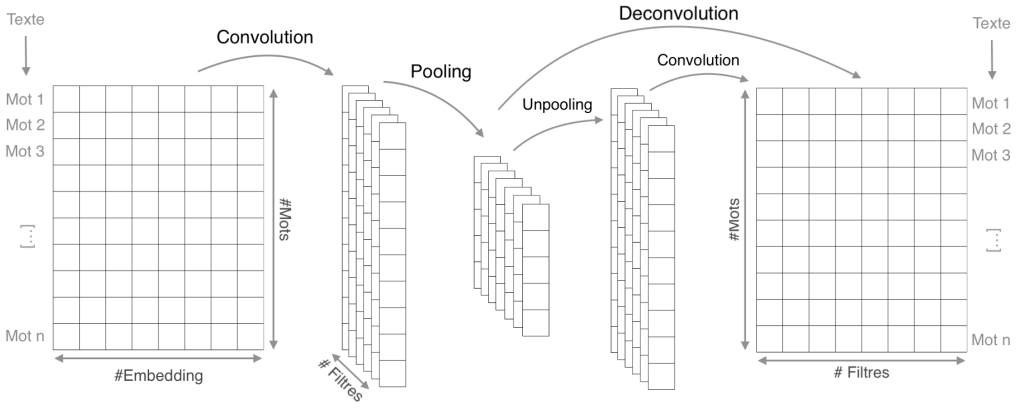


FIGURE 3.10 – Déconvolution par convolution transposée

Cette méthode a l'avantage de pouvoir être adaptée à n'importe quelle architecture *convolutionnelle*. Ici la convolution transposée garantit toujours un retour à l'espace vectoriel de départ interprétable grâce à l'*unpooling* même dans le cas d'une *convolution* multiple qui compresse successivement l'information avec plusieurs *convolutions* et des tailles de filtres différentes. Cette méthode de *déconvolution* permet de décompresser l'information pour obtenir une approximation de l'information apprise par le réseau. L'inconvénient majeur est que l'information obtenue reste une approximation dont la précision peut jouer significativement sur le risque de surinterprétation des résultats. Pour éviter cet effet il existe une autre méthode qui repose sur une astuce simple qui rend la *convolution* interprétable par l'humain sans avoir à ajouter de couche supplémentaire.

3.3.2. Méthode 2 : architechture contrainte

La convolution est un algorithme paramétrable qui peut être ajusté suivant les besoins. Lorsque le besoin principal de l'utilisateur est l'interprétation des données et non la performance, il est possible de réaliser une convolution sans compresser l'information. Par défaut, la convolution opère depuis le premier bord du texte (le premier mot) puis le balaye de mot en mot jusqu'au dernier. La manière de traiter les bords du texte s'appelle le *padding* et la taille du déplacement s'appelle le *stride*. Le modèle standard appliqué sur le texte utilise un *padding* dit *normal* et un *stride* de 1 (mot à mot) (figure 3.11).

Ce comportement peut être modifié pour pouvoir reconstruire en sortie une nouvelle image de taille identique à celle de départ. Le balayage des filtres doit pour cela être étendu en dehors des limites du texte. Autrement dit, il faut permettre aux filtres de dépasser les

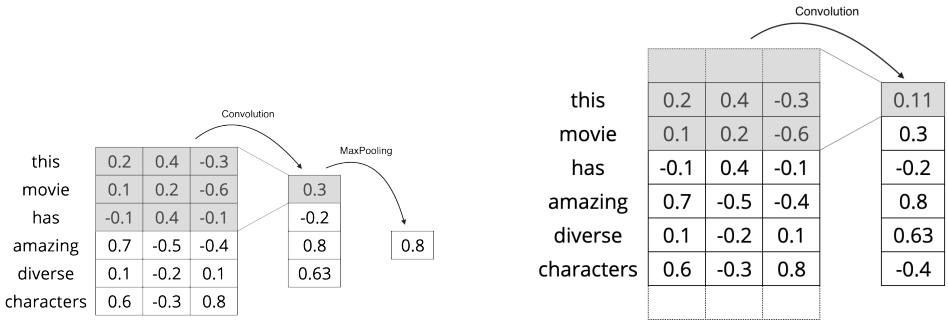


FIGURE 3.11 – Déconvolution 2 : *padding normal* à gauche et *padding same* à droite.

bords de la matrice. C'est l'algorithme *padding same* (figure 3.11). Avec cette méthode, les premiers mots bénéficient seulement d'une partie du contexte, mais si on conserve le *stride* par défaut de 1 mot, la sortie de ce type de *convolution* a l'avantage de proposer une sortie directement lisible et interprétable puisqu'elle ne compresse pas les données (à noter que dans ce cas précis, aucune couche de *pooling* ne doit être ajoutée). Cette solution simplifie grandement l'opération de *déconvolution* qui se résume alors à la simple lecture de la couche de sortie de cette *convolution* particulière. Néanmoins il faut noter que l'absence de compression d'information lors de la convolution peut avoir un impact négatif sur les performances (taux d'*accuracy*) du modèle final. Cette baisse des performances peut être néanmoins compensée en augmentant le nombre de neurones dans les couches denses suivantes (sur une ou plusieurs couches) pour augmenter leur capacité à traiter des tailles de données plus importantes.

3.4. Interprétabilité

Quelle que soit la méthode utilisée, la *déconvolution* permet donc d'extraire l'information encodée dans les couches cachées de la convolution. Pour comprendre la nature de cette information et pouvoir véritablement l'utiliser en ADT, une idée simple consiste à projeter les données obtenues par *déconvolution* sur une image en deux dimensions. Chaque pixel de l'image correspond à la cellule d'une matrice qui croise en ligne les mots de la séquence et en colonne les dimensions de la représentation vectorielle. L'intensité (le niveau de gris) de chaque pixel est réglée pour correspondre au niveau d'activation de chaque neurone associé. Cette technique de visualisation illustre les premières observations du **TDS** (section 3.2.4) et permet de confirmer que cette méthode est un bon moyen pour retourner au texte lorsqu'on utilise un réseau à convolution (figure 3.12).

L'interprétation et l'utilisation d'une déconvolution semblent évidentes : certaines lignes, et donc certains mots sont en moyenne plus activés que d'autres. L'hypothèse de VANNI, DUCOFFE et al. 2018 se vérifie donc, et il s'agit là des saillances du texte apprises par le modèle, observé dans les couches cachées du réseau de neurones. Cette contribution est majeure pour la compréhension de la *convolution* et la description des textes par le *deep learning* puisque c'est là un moyen d'interpréter les choix du modèle en associant un poids à chacun de mots en fonction des activations des neurones du réseau. Ce poids que nous avons appelé *Text Deconvolution Saliency (**TDS**)* correspond donc à la somme des

FIGURE 3.12 – Extraction brute des données de déconvolution

activations de l'*embedding* de chaque mot.

Plus formellement, cette architecture permet de classer un segment de texte d_i en proposant des indices d'interprétations $TDS(d_im)$ pour chacun des m-ième tokens. d_i est représenté dans le réseau par matrice réelle avec M lignes où M est le nombre de tokens de d_i et D colonnes où D est la taille de l'*embedding*. Le m-ième token de d_i correspond donc à la m-ième lignes de la matrice que l'on note d_{im} un vecteur $\in \mathbb{R}^D$. La couche de *déconvolution* assigne alors à chaque d_{im} un nouveau vecteur de même taille noté $x_{im} \in \mathbb{R}^D$. Ce vecteur étant sensible au contexte de d_{im}. Le calcul du **TDS** pour chaque token s'écrit alors :

$$TDS(d_{im}) = \sum_{d=1}^{D} x_{imd} \tag{3.1}$$

Où le nombre réel x_{imd} est le d-ième élément de x_{im}.
Les premiers résultats du **TDS** sont encourageants. Chaque ligne en surbrillance dans la figure 3.12 correspond à un mot auquel le réseau a accordé plus d'importance qu'un autre pour prendre sa décision. Au niveau des calculs, cela signifie que le taux d'activation y_i de chacun des neurones qui correspondent à ce mot est plus élevé et qu'il contribue plus fortement à la probabilité de sortie.

Linguistiquement cela se traduit par une mise en valeur de certains mots, expressions ou *patterns* que le réseau utilise pour prendre sa décision (classification). Le **TDS** est donc un nouvel outil qui attribue un poids à chaque mot, basé sur un calcul non linéaire (lié aux fonctions d'activation de chaque neurone, elles-mêmes non linéaires) et que la statistique classique ne peut engendrer. Avec la convolution, une même forme graphique prend un poids différent suivant son contexte, là où la statistique ne propose qu'un seul score de spécificité (ou *z-score*) pour chacune des classes ou qu'un profil cooccurrentiel général unique dans le corpus.

Cette différence est vraisemblablement à l'origine des écarts observés entre *deep learning* et statistiques en tâche de classification. Il est possible de comparer d'ailleurs les performances de notre modèle à plusieurs solutions de classification statistiques standard

connues comme la régression logistique (LOGIT) ou encore les *random forest* (Rforest). Soumise à quelques corpus d'attributions d'auteurs qui croisent des langues, des tailles et des nombres de classes différentes, la représentation du texte engendrée par la convolution semble plus efficace pour discriminer nos classes (tableau 3.2).

langue	nb classes	vocabulaire	occurrences	LOGIT	Rforest	deep learning
français	8	46978	2 738 652	81%	47%	93%
anglais	11	33279	1 815 839	75%	50%	90%
latin	22	129348	2 035 176	95%	46%	96%

TABLE 3.2 – Benchmark statistique vs *deep learning*

Ce stade de l'analyse descriptive *deep learning* est donc très prometteur. L'IA offre visiblement de nombreuses lectures possibles quand à ses choix et sa représentation des textes. Néanmoins même si cette nouvelle mesure semble rendre compte de saillances textuelles fortes et contrastives, rien n'indique qu'elles ont véritablement contribué ou au contraire contrarié le choix final de la classification. En effet, un **TDS** important pour un mot ne veut pas forcément dire qu'il appartient à la classe choisie par le modèle. L'exemple qui suit va permettre de rendre compte de ce phénomène et des limites du présent calcul du **TDS**.

3.5. Limites du TDS

La convolution produit une abstraction des données pertinente pour la tâche de classification demandée, ce mécanisme se traduit par la détection de marqueurs linguistiques forts qui soulignent les contrastes d'un partitionnement de corpus. Cependant, quelle que soit la classe associée à un segment de texte, le **TDS** montrera l'ensemble des marqueurs, même ceux qui ont inversement contribué à l'attribution de la classe.

Pour l'exemple suivant le corpus utilisé réunit l'ensemble des discours officiels des présidents américains depuis J. F. Kennedy jusqu'à D. Trump, soit environ 4 millions de mots, qui a été soumis au calcul des passages-clés (voir Section 5.2). Dans le cas du discours de D. Trump un des segments les plus significatifs pour l'IA est l'extrait suivant :

> *[...] neighborhoods for their families , and good jobs for themselves . These are just and reasonable demands of righteous people and a righteous public. But for too many of our citizens , a different reality exists : Mothers and children trapped in poverty in our inner cities ; rusted-out [...]*

(D. Trump, Discours d'investiture le 20 janvier 2017, Washington, D.C.)

Le modèle est sensible visiblement à l'enchaînement *righteous people ... righteous public* qui regroupe naturellement des spécificités fortes du président américain (caractérisant là un discours populaire classique). Mais il y a aussi le mot *poverty* mis en évidence par le **TDS** qui au contraire est sous-représenté chez le président américain avec un *z-score* de -7.9 (figure 4.11).

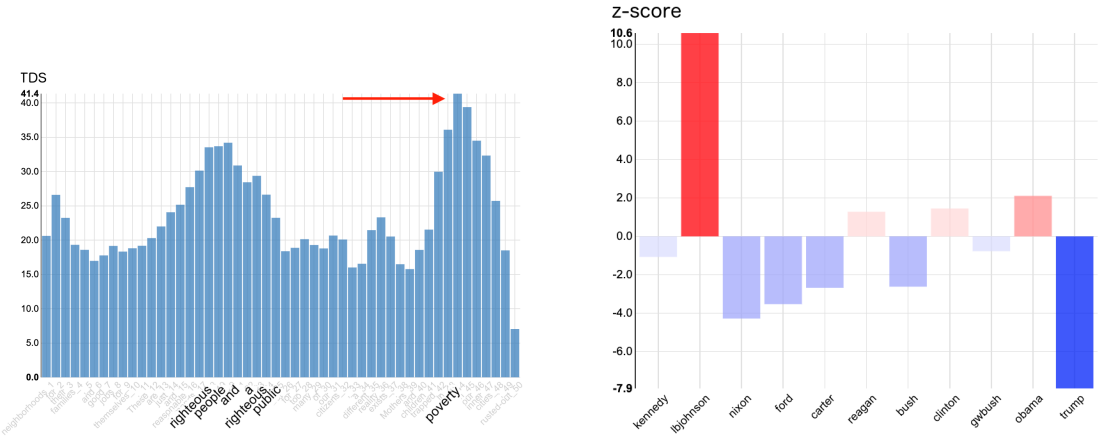


FIGURE 3.13 – Comparaison du **TDS** et du z-score de *poverty* dans le discours présidentiel américain

Ce résultat contre-intuitif doit donc être analysé délicatement. La convolution est sensible au contexte, or même en ajoutant les mots qui précèdent ou qui suivent *poverty* dans le calcul des spécificités aucun segment ici ne semble contribuer à D. Trump. D'ailleurs l'expression exacte *trapped in poverty* n'est pas unique à D. Trump, elle a déjà été prononcée au moins une autre fois par Lyndon B. Johnson dans le discours d'inauguration de l'université de Howard le 4 juin 1965. Le *deep learning* donnerait-il une information erronée sur le président Trump ? Comme l'indique la statistique (figure 4.11 droite), ce passage du segment est un marqueur fort du corpus des présidents américains. Mais absolument pas de l'actuel président. C'est même plutôt l'inverse, le thème de la pauvreté n'est statistiquement pas un des thèmes favoris de D. Trump, mais plutôt celui de Lyndon B. Johnson qui l'a utilisé 117 fois dans l'ensemble de ses discours officiels, soit un z-score de 10.6. Le **TDS** a donc bien repéré un marqueur linguistique fort du discours présidentiel américain utile pour la classification des textes, mais qui a dû forcément ici contribuer inversement à l'attribution de ce passage à D. Trump.

Pour appuyer notre raisonnement il est possible aussi de comparer le **TDS** avec un autre système d'aide à l'interprétation, le *Local Interpretable Model-Agnostic Explanations* (LIME - voir section 4.3). Cette méthode largement diffusée dans la communauté du *machine learning*[10] permet d'interpréter n'importe quel modèle de classification en se basant simplement sur les entrées et sorties. En d'autres termes en ajoutant du bruit dans les données (en remplaçant des mots) et en observant les effets sur les probabilités en sortie (voir section 4.3). Cette méthode donne des résultats robustes sous forme de spécificité pour chaque mot, insuffisants pour les besoins d'analyses linguistiques de corpus, mais qui donne de solides indicateurs pour la compréhension générale des prises de décisions de notre modèle. Et sur notre exemple, LIME confirme bien notre intuition suggérée par le z-score. Parmi les mots qui font le plus baisser la probabilité pour D. Trump il y a bien *poverty* en premier (figure 3.14).

10. Ici il est bien question de machine learning en général et non juste de *deep learning* car LIME sert aussi à interpréter des classifieurs linéaires classiques comme les SVM et autres random forest

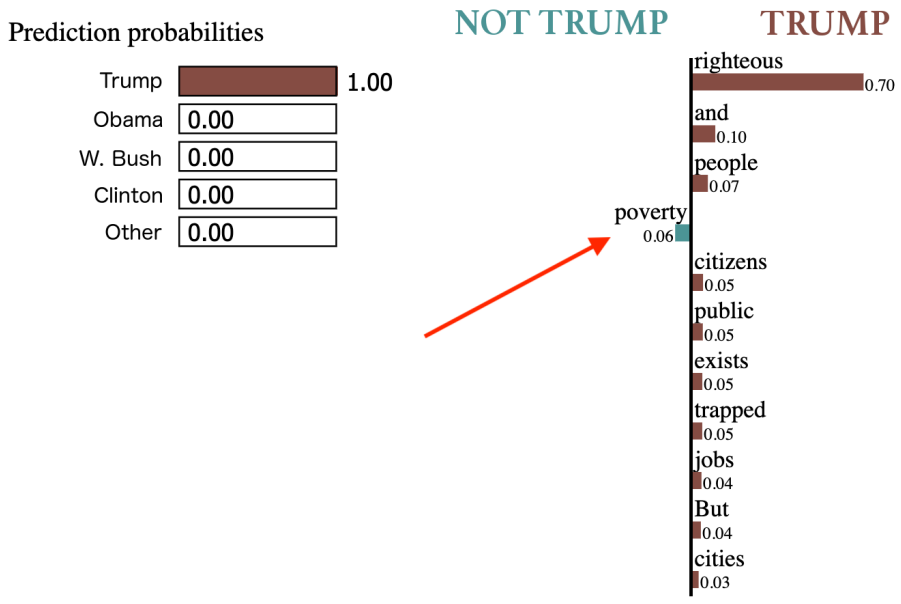


FIGURE 3.14 – Application de LIME sur l'extrait du discours de D. Trump.

Le premier calcul du **TDS** se révèle donc assez naïf et manque de précision pour permettre une véritable utilisation en ADT. Nous allons voir maintenant dans une étude qui généralise le problème aux RNNs, une technique qui permet d'ajuster ce calcul et prendre en compte la classe observée en sortie pour aboutir à un **TDS** plus robuste et efficace pour la linguistique textuelle.

Chapitre 4

Text Class Activation Map : T-CAM

Ce chapitre inédit présente l'algorithme utilisé pour calculer le TDS pondéré (wTDS). Cette version amélioré du TDS a été initialement présenté dans Vanni, Corneli, *D.* Longrée *et al. 2020a. Le détail du calcul présenté ici est à paraitre (*Vanni, Corneli, *F.* Precioso *et al. 2022)*

4.1. Introduction

L'un des principaux défis actuels des techniques d'apprentissage profond réside dans les difficultés à expliquer ou même à interpréter le comportement des classificateurs. Les premiers efforts pour expliquer les réseaux de neurones profonds se sont concentrés sur les réseaux de neurones convolutifs (CNN), sur la visualisation des filtres appris et sur la mise en évidence des parties spatiales de toute image impliquée dans la classification (Zeiler et Fergus 2014).

Toujours dans le contexte de l'analyse d'image, Zhou et al. 2016 ont introduit la notion de *Class Activation Mapping* (CAM) qui attribue un score de classification pour chaque classe à chaque emplacement de l'image d'entrée. Le CAM adopte les sommes pondérées des caractéristiques convoluées pour chaque pixel d'entrée. Malgré sa capacité à détecter des caractéristiques discriminantes *pour chaque classe*, certaines critiques ont été soulevées à l'encontre de CAM, en particulier : i) il est nécessaire d'introduire une couche de regroupement (*MaxPooling* ou *AveragePooling*) dans l'architecture originale du CNN, modifiant ainsi la structure du classificateur. Ce choix peut diminuer la précision de classification du CNN et le rend impossible à travailler avec des images à haute résolution (Selvaraju et al. 2017). ; ii) une seule couche entièrement connectée est adoptée pour terminer le réseau, alors que plusieurs architectures adoptent en général plusieurs de ces couches avant la sortie du réseau.

Certaines tentatives pour surmonter ces problèmes ont été faites en introduisant Grad-CAM (Selvaraju et al. 2017) et plus tard GradCAM++ (Chattopadhay et al. 2018). Afin d'obtenir un score pondéré des marqueurs de la convolution, au lieu d'utiliser simplement les poids de la dernière couche *Dense* du CNN (comme le fait CAM), ces méthodes calculent toutes deux la moyenne des dérivées partielles de la couche de sortie par rapport aux emplacements des marqueurs dans le réseau. Cependant, il a été récemment souli-

gné que l'utilisation de dérivées partielles peut entraîner de graves perturbations pendant l'entraînement du réseau (saturations et erreurs d'attributions). De plus, une limitation partagée des versions existantes de CAM est qu'elles ne fonctionnent pas avec des architectures autres que CNN (par exemple RNN). Cette limitation pourrait être particulièrement pertinente dans l'exploration de texte.

Nous proposons donc ici une extension de CAM travaillant avec des données textuelles (T-CAM). Ainsi, T-CAM met en évidence les régions pertinentes dans les paragraphes de texte (c'est-à-dire les marqueurs linguistiques) en activant chaque classe. De plus, T-CAM ne nécessite pas l'intégration d'une couche de *Pooling* dans le réseau, il permet d'intégrer davantage de couches finales *Dense* dans le réseau, et peut être généralisé à plusieurs architectures, y compris les RNNs. Avant de plonger dans les détails de T-CAM (section 4.4), nous présentons dans la section suivante des travaux connexes, certains visant à étendre les techniques d'explications existantes de l'analyse d'images à la classification de textes, et d'autres, entièrement nouveaux.

4.2. Context

Depuis les travaux de R. Collobert et Weston 2008, les CNNs ont été adoptés pour plusieurs tâches d'analyse de textes telles que l'étiquetage des catégories grammaticales (*Part-Of-Speech*), la tokenisation, la lemmatisation, la reconnaissance d'entités nommées et l'étiquetage sémantique. De nombreux chercheurs ont aussi largement utilisé les CNNs pour d'autres tâches telles que la modélisation de textes (Nal Kalchbrenner, Edward Grefenstette et Phil Blunsom 2014) ou la classification de phrases (Kim 2014b).

Bien que les CNNs ne soient pas la seule architecture profonde disponible dans le domaine de la fouille de textes, il a été remarqué qu'ils présentent plusieurs avantages par rapport aux réseaux neuronaux récurrents (RNN, en particulier LSTM et GRU) lors de la reconnaissance de phrases clés (Yin et al. 2017) ou par rapport aux réseaux basés sur les *transformer* (Devlin et al. 2019) lorsqu'on considère l'interprétabilité du réseau.

Un autre avantage des CNNs est leur capacité à prendre en compte les dépendances à long terme dans les phrases grâce à la convolution. En effet comme il à déjà été noté dans la section 3.2, les CNNs semblent être plus robustes que les RNNs au problème de la disparition du gradient (*vanishing gradient problem*), ils pourraient donc être capables de détecter des liens entre différentes parties d'une phrase (Dauphin et al. 2017 ; Wen et al. 2017 ; Adel et Schütze 2017). Cette propriété est cruciale, car il a été démontré que des dépendances à long terme émergent dans les données réelles (Li et al. 2015). Afin d'étudier ces dépendances ainsi que des marqueurs linguistiques complexes, il est nécéssaire d'utiliser des outils permettant d'expliquer la classification d'un modèle.

4.3. LIME

À cet égard, le *Local Interpretable Model-agnostic Explanations* (Ribeiro, Singh et Guestrin 2016) constitue une contribution importante. L'idée de base de LIME est de fournir une approximation à tout classificateur complexe (par exemple CNN) par un

classificateur plus simple (linéaire) dans le voisinage d'un point d'apprentissage x_i (un morceau de texte donné à classer). Une représentation simplifiée $\tilde{x_i}$ de x_i est adoptée, et N points dans le voisinage de $\tilde{x_i}$ sont échantillonnés uniformément et utilisés pour minimiser une distance entre le classifieur original et le plus simple. Une fois que le classificateur plus simple est entrainé, il peut être utilisé pour évaluer la contribution positive (ou négative) de chaque caractéristique à la classification aussi facilement que dans les modèles linéaires. Pouvant fournir des explications pour n'importe quel classificateur, la robustesse de LIME a récemment été évaluée d'un point de vue statistique (Garreau et Luxburg 2020), également pour des données textuelles (Mardaoui et Garreau 2020). Bien qu'étant la référence des outils d'aide à l'inteprétation de classifieurs, LIME nécessite d'échantillonner N voisins de chaque point d'apprentissage et d'évaluer le classifieur pour chacun de ces points. Ceci est rapidement prohibitif en termes de calcul, en particulier de grandes quantités de données.

4.4. Architecture générale

Cette section décrit comment un score d'activation dépendant de chaque classe (T-CAM) peut être obtenu à partir de plusieurs architectures. Pour des raisons de clarté, nous commençons par un exemple simple avec un réseau neuronal à une seule couche cachée.

La figure 4.1 illustre un classificateur simple à base de réseaux de neurones pour l'analyse de sentiments. Cette architecture est composée d'une couche d'*Embedding* puis d'une couche *Dense* (couche cachée) et une couche de sortie à deux neurones, étant donné le problème de classification binaire considéré. Même si la couche d'*Embedding* peut s'appuyer sur différents modèles tels que *fast-Text* (Bojanowski et al. 2017 ; Joulin, Grave et P. B. T. Mikolov 2017), *Word2Vec* (T. Mikolov et al. 2013) ou Glove (Pennington, Socher et C. Manning 2014), le modèle permet ici un ajustement (entrainement) de cette représentations vectorielles des mots pendant l'apprentissage (le choix de la méthode de préapprentissage n'est donc pas critique).

Après avoir entrainé le modèle sur le jeu de données ACL-IMDB (plus de détails dans la section 4.7), nous testons le modèle avec la phrase *Great movie with bad end.* Cette phrase est ambigüe en raison de la présence des deux mots *great* et *bad*. Cependant, notre modèle classe ce segment comme positif. Notre objectif est de distinguer les tokens utilisés par le réseau neuronal pour classer la phrase comme *positive* de ceux qui suggèreraient un étiquetage *negatif*.

Une première approche pour obtenir des scores d'activation pour chaque token adopterait l'idée du **TDS** suggérée dans la section 3.2 (Vanni, Ducoffe et al. 2018). En effet, en additionnant les entrées de chaque vecteur d'enchâssement $x_{im} \in \mathbb{R}^D$ met en évidence les mots pertinents dans d_i utilisés par le modèle pour classer ce paragraphe (voir Figure 4.2).

À noter que les scores les plus élevés sont obtenus par les mot *great* et *bad*, ce qui est juste dans le contexte de l'analyse de sentiments : ces deux mots sont importants pour la prédiction. Cependant, comme ils renvoient à des sentiments opposés, le score de ces deux

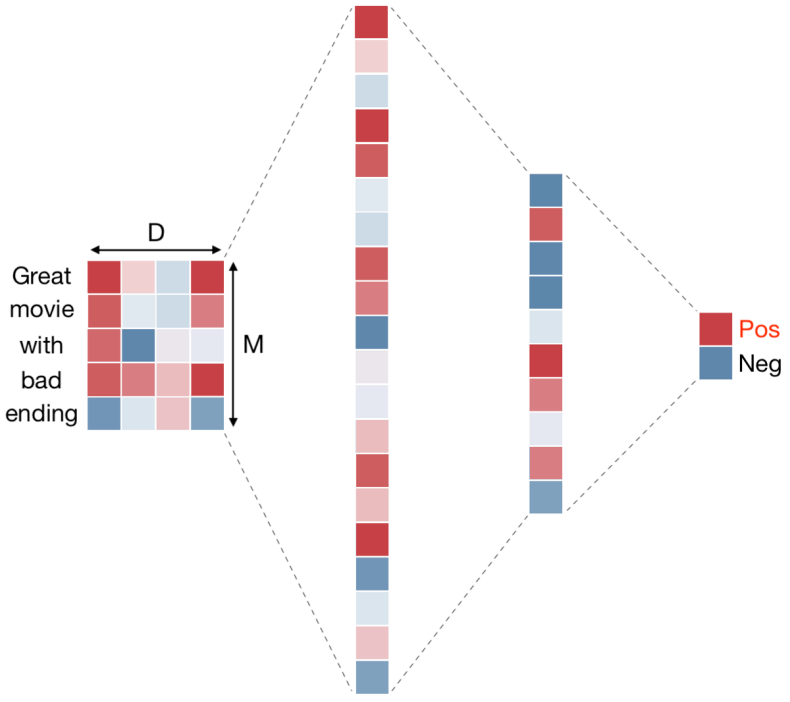


FIGURE 4.1 – Un réseau *fully-connected* simple pour l'analyse de sentiments. En bleu et rouge, respectivement, on peut voir l'activation positive et négative de chaque neurone.

mots doit être pondéré quelque part dans le réseau neuronal pour qu'il puisse distinguer leur participation à chaque classe.

Afin d'obtenir des scores positifs et négatifs, et en s'inspirant des résultats de CAM et Grad-CAM pour les images, nous introduisons un score d'activation pour chaque mot dépendant de la classe observée (T-CAM). L'idée principale de T-CAM est d'utiliser les poids des dernières couches entièrement connectées (*Dense*) du réseau neuronal pour ajuster les représentations des mots.

En effet, les couches *Dense* transforment l'abstraction des données faite par le réseau (l'*Embedding* dans ce cas) en un vecteur de sortie menant à la prédiction de classe, généralement via une fonction d'activation de type *softmax*. Lorsque les couches *Dense* sont appliquées à l'ensemble des données, le vecteur de sortie conduit à la prédiction de la classe pour l'ensemble du paragraphe. Inversement, si ces couches n'agissent que sur un mot/vecteur (une sous partie des données), elles produisent alors un score correspondant à la contribution de ce mot pour chaque classe (Figure 4.3). Plus formellement, désignons par y_i la sortie/vecteur correspondant au paragraphe d'entrée texte d_i, avant application de la fonction *softmax*. Alors :

$$y_i = d + C\left(\text{relu}\left(b + Ax_i\right)\right) \tag{4.1}$$

où x_i à la sortie de l'*Embedding* sous forme de vecteur (*Flatten*), $A \in \mathbb{R}^{E \times DM}$, $b \in \mathbb{R}^E$,

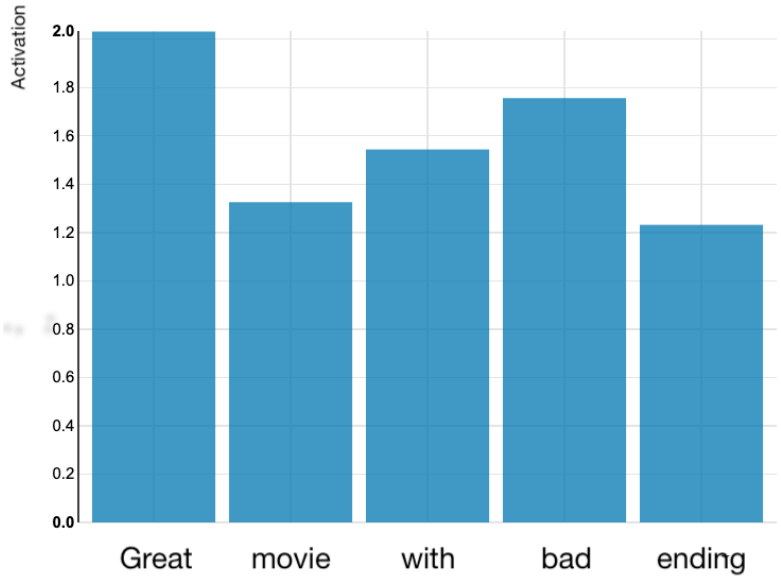


FIGURE 4.2 – Scores d'activation obtenus en additionnant les composantes mot/vecteur.

$C \in \mathbb{R}^{E \times K}$ et $d \in \mathbb{R}^K$, où E est la taille de l'avant-dernière couche et $K = 2$ dans la figure 4.1. Ensuite, pour chaque *token* d_{im}, le score T-CAM est défini comme suit :

$$\text{T-CAM}(d_{im}) := d + C\left(\text{relu}\left(b + A_m x_{im}\right)\right) \tag{4.2}$$

où $A_m \in \mathbb{R}^{E \times D}$ est la sous-matrice de A obtenue en sélectionnant toutes les lignes et les D colonnes de la $(D(m-1)+1)$-ième à la $(D(m-1)+D)$-ième.

Dans l'exemple de la Figure 4.1, l'équation ci-dessus identifie un vecteur à deux entrées, correspondant à la contribution du mot d_{im} à chaque classe. À noter que le *softmax* n'est pas appliqué au côté droit de l'Eq. (4.2) ce qui permet d'obtenir aussi bien des scores positifs que négatifs.

Lors du calcul du score T-CAM pour l'exemple de test précédent (Figure 4.1), la contribution du mot *bad* est négative pour la classe *positif* et vice versa (voir Figure 4.4). Nous verrons dans la section 4.7 que T-CAM est encore plus pertinent lorsqu'il est utilisé pour expliquer les décisions d'un modèle dans une classification à classes multiples.

4.5. Modèle appliqué aux CNNs

T-CAM peut facilement être calculé pour des architectures plus sophistiquées qu'un simple réseau de neurones avec une seule couche *fully-connected*. En effet, la seule condition pour que l'équation (4.2) tienne est un vecteur caractéristique x_i qui représente le texte en entrée avec la possibilité de distinguer chaque token (mots). Dans l'exemple précédent, x_i était la sortie de l'*Embedding* mais, plus généralement, il pourrait s'agir de la sortie de *n'importe quelle* couche interne.

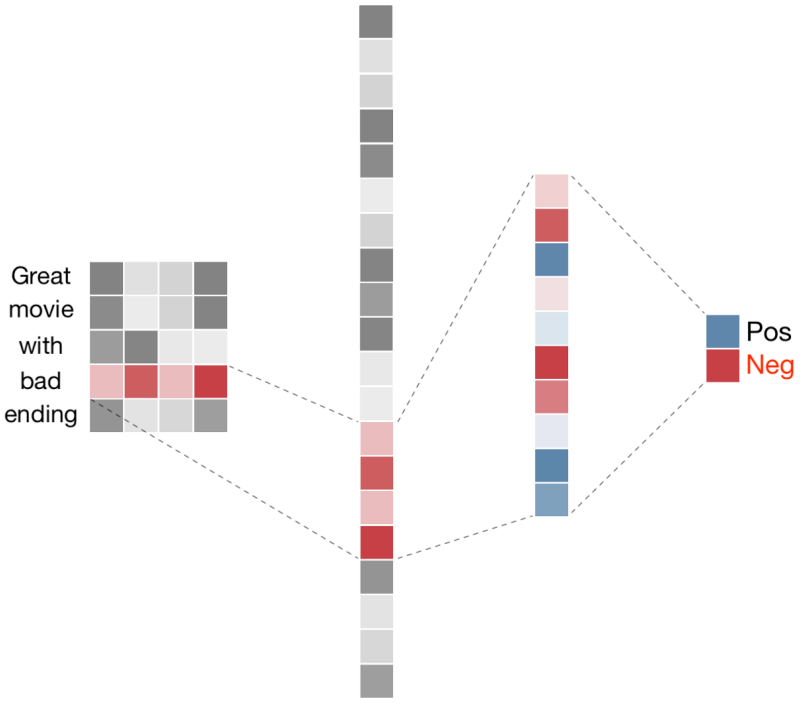


FIGURE 4.3 – Calcul du score T-CAM pour le mot *bad*.

Avec le **TDS** par exemple (VANNI, DUCOFFE et al. 2018), la dimension du texte (M tokens) est obtenue par *unpooling* de la sortie de la convolution et en appliquant une convolution transposée (c'est-à-dire par déconvolution). Cette méthode donne une bonne approximation de ce qui est appris par les couches convolutionnelles et peut être utilisée comme entrée de l'algorithme T-CAM.

Une autre approche consiste à utiliser un *padding same* dans les couches convolutives, supprimant ainsi la couche de *Pooling* pour conserver la taille des données textuelles d'entrée. La figure 4.5 montre un exemple d'une telle architecture avec quatre filtres de trois mots chacun. La sortie de la couche convolutionnelle est une sorte de nouvel *Embedding* qui peut être utilisé par T-CAM pour calculer les scores d'activation des mots pour chaque classe.

Soulignons que dans le CAM (ZHOU et al. 2016) original (pour l'analyse d'image) un *Pooling* (moyen ou maximum) global doit être intégré dans l'architecture du réseau, inversement T-CAM utilise directement la couche de sortie vectorisée (*Flatten*) pour calculer les scores d'activation. C'est également la raison pour laquelle plus de couches *fully-connected* sont autorisées pour le T-CAM alors qu'elles ne le sont pas pour le CAM standard. Cependant, comme pour CAM, T-CAM impose des contraintes à l'architecture du réseau, puisqu'une matrice de taille M ligne (correspondant au nombre de tokens) est requise.

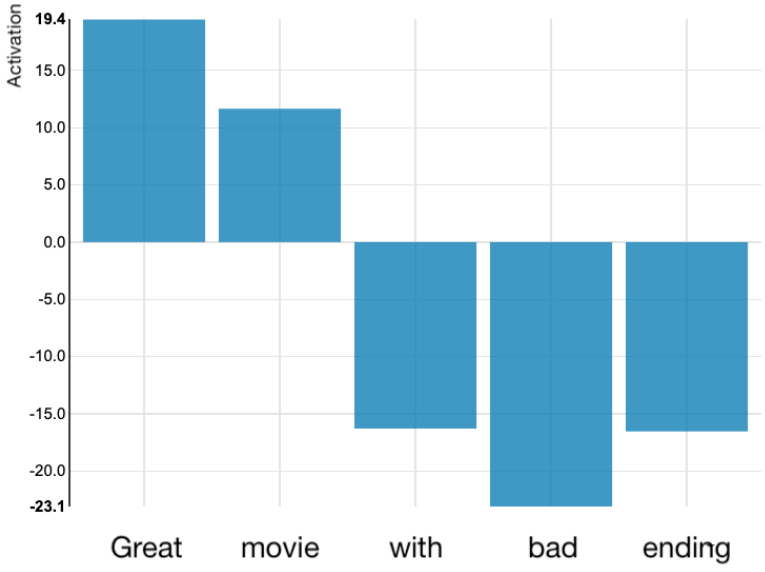


FIGURE 4.4 – Score T-CAM pour la classe *positive*.

4.6. Modèle appliquée aux RNNs

Les LSTM ont été introduits par HOCHREITER et SCHMIDHUBER 1997 pour capturer les dépendances à long terme dans les séquences de texte, en tant qu'alternative aux architectures RNN plus simples. Les architectures LSTM et GRU (CHO et al. 2014) sont utilisées pour plusieurs tâches d'apprentissage automatique telles que la modélisation du langage, la reconnaissance vocale ou la prédiction de séries temporelles.

Afin d'obtenir des explications à partir des RNN (y compris les LSTM et les GRU), une approche raisonnable consiste à examiner les couches cachées des cellules des RNNs pour déduire l'importance de chaque token dans une séquence donnée. Une formalisation bien connue de cette idée est le mécanisme d'attention (VASWANI et al. 2017), qui s'appuie sur une couche *Time Distributed Dense (TDD)* pour fournir à chaque token d'entrée un score d'activation (qui correspond à l'attention). Intuitivement, le score identifie l'importance du token pour réaliser la tâche. Ainsi, la visualisation des scores d'attention fournit des informations intéressantes pour interpréter une prédiction RNN.

Cependant, nous affirmons que, dans le contexte de la classification de textes, cette interprétabilité peut être améliorée en appliquant l'algorithme de T-CAM à l'attention, comme le montre la figure 4.6. La dernière cellule RNN (LSTM ou GRU) produit une matrice en $\mathbb{R}^{M\times E}$, où M est le nombre de tokens dans le paragraphe d'entrée et E désigne ici le nombre de neurones dans la cellule RNN cachée. Après l'expansion et la transposition du vecteur de la couche TDD de sortie, une autre matrice en $\mathbb{R}^{M\times E}$ est obtenue et nous effectuons un produit élément par élément entre les deux matrices pour garder les dimensions fixes, maintenant ainsi une relation avec les tokens d'entrée. Comme on peut le voir sur

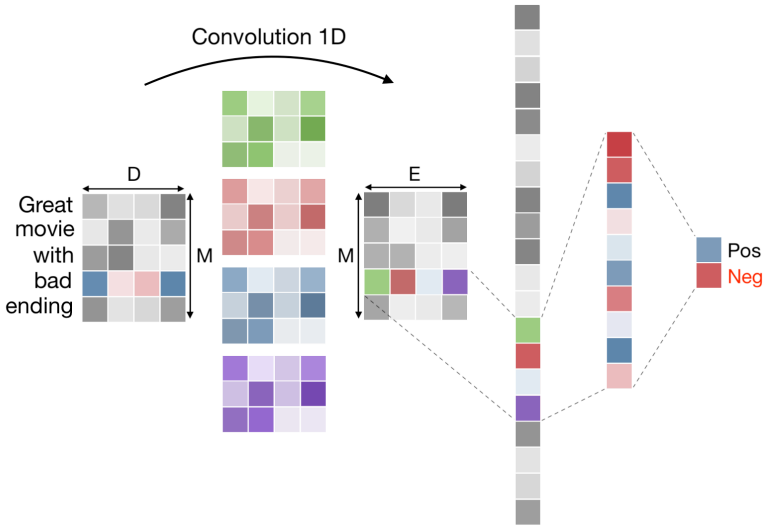


FIGURE 4.5 – T-CAM applied on CNN.

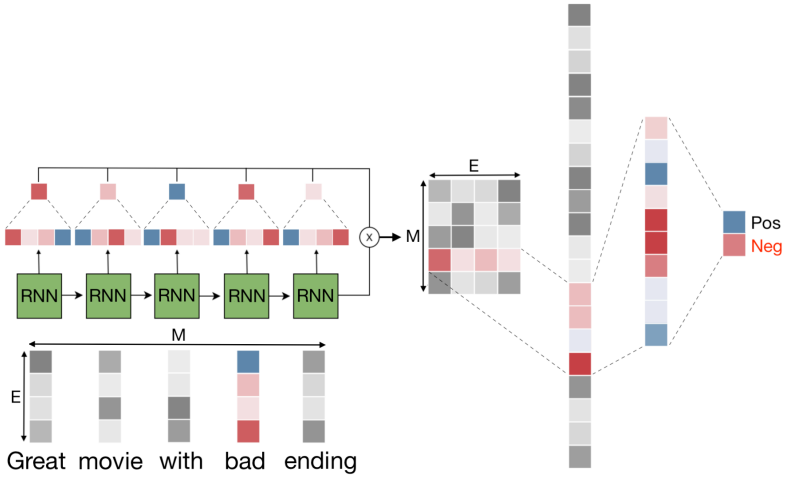


FIGURE 4.6 – T-CAM applied on RNN.

la Figure 4.6, les dernières étapes pour produire les scores T-CAM sont analogues à celles que nous avons effectuées avec les architectures précédentes.

Nous soulignons enfin que cette architecture est un moyen simple de combiner le T-CAM avec des RNN+Attention. Des architectures récurrentes plus sophistiquées pourraient bien sûr être développées dans le cadre de recherches futures.

4.7. Expériences

Cette section vise à évaluer la qualité des marqueurs linguistiques discriminants extraits par T-CAM.

4.7.1. protocole

Comme nous l'avons vu, un avantage majeur de T-CAM est qu'il peut être calculé pour plusieurs architectures. Dans cette section, nous adoptons deux modèles différents : i) Le CNN de la Figure 4.5, composé d'une couche d'*Embedding*, d'une couche convolutionnelle à une dimension (avec fonction d'activation *relu*) et de deux couches entièrement connectées ; ii) Le RNN de la Figure 4.6 composé d'une couche d'*Embedding*, d'une cellule LSTM avec couche d'attention (TDD avec fonction d'activation *tanh*) et de deux couches entièrement connectées.

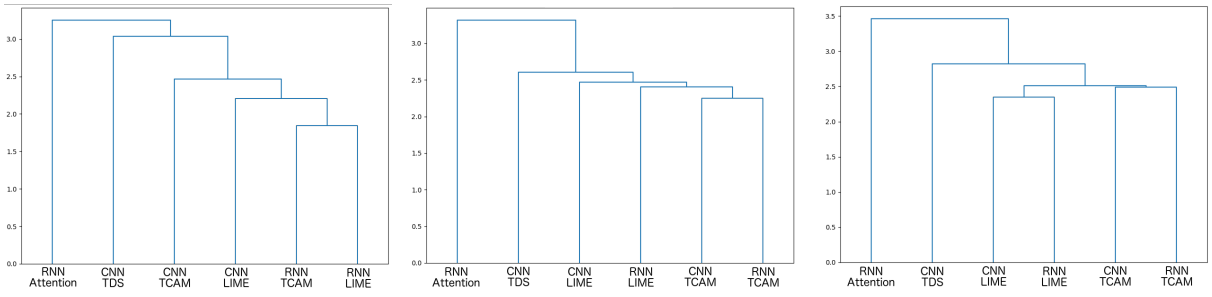
Les marqueurs extraits avec T-CAM sont comparés à ceux obtenus via des approches alternatives telles que **TDS** pour les CNNs, les couches d'attention pour les RNNs et LIME (pour les CNNs et les RNNs). En raison de sa large utilisation dans la communauté de l'apprentissage automatique ainsi que des résultats mentionnés dans la section 4.3, prouvant sa robustesse statistique, LIME est notre principal concurrent. En particulier, lors de la détection des caractéristiques qui contribuent positivement à la prédiction d'une classe, nous considérons que faire aussi bien que LIME est le mieux que nous puissions faire.

T-CAM présente tout de même deux caractéristiques différentes par rapport à LIME qui le rend plus performant. Premièrement, il est de loin moins coûteux en calcul, puisqu'il ne nécessite aucune étape d'échantillonnage. Deuxièmement, T-CAM détecte simultanément les scores d'activation de chaque token pour toutes les classes, alors que LIME doit binariser la tâche de classification, même en classification multi-classes.

Afin de rendre ces expériences reproductibles, nous utilisons des ensembles de données bien connus pour trois tâches de classification différentes : 1) l'analyse de sentiments, 2) la modélisation de thèmes et 3) l'attribution de paternité.

1. **ACL-IMDB** composé d'un ensemble d'entrainement et de test de 25 000 critiques de films chacun, étiquetés comme positifs ou négatifs. Ce corpus est une référence pour l'analyse de sentiments avec une classification binaire. Sur le jeu de données de test, nos modèles obtiennent une précision de 85,45% (CNN) et de 83,19% (RNN).
2. **AG-News** est un corpus d'articles qui concerne la classification par *topic*. Il compte 120 000 échantillons pour l'entrainement plus 7 600 pour le test et s'étend sur 4 classes (*Business*, *SciTech*, *Sports*, *World*). Sur le jeu de données de test, notre CNN obtient une précision de 95,36%, 94,82% pour le RNN.
3. **Reuter_50_50** (C50) pour la classification des auteurs. Nous nous concentrons ici sur un sous-ensemble du RCV1 (*Reuters Corpus Volume 1*), avec 2 500 textes pour l'ensemble de données d'entrainement et 2 500 textes pour l'ensemble de test, dans les deux cas 50 textes ont été sélectionnés pour chaque auteur. Sur le jeu de données de test, notre modèle obtient une précision de 71,11% avec le CNN et de 78% avec le RNN.

Comme nous l'avons vu dans la section 4.4, l'utilisation de T-CAM nécessite l'application de contraintes architecturales au réseau de neurones. Comme indiqué dans (Zhou



(a) ACL-IMDB. (b) AG-News. (c) C50.

FIGURE 4.7 – Classification hiérarchique basée sur la matrice de distance euclidienne entre les vecteurs de score pour les trois ensembles de données.

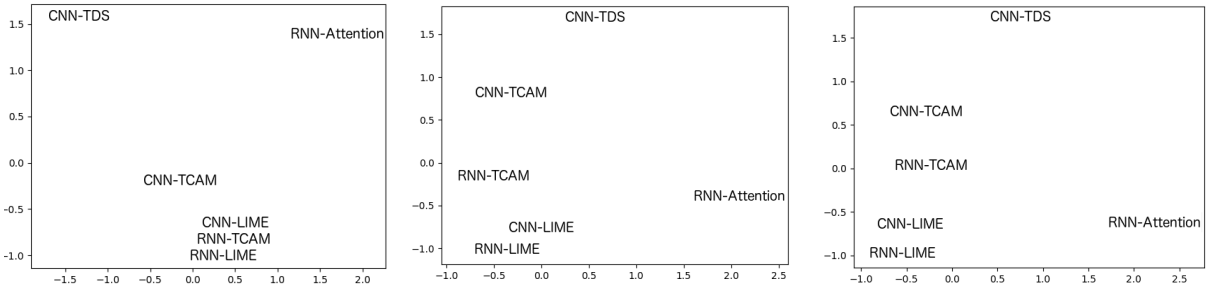
et al. 2016), bien que les contraintes architecturales puissent légèrement affecter les performances du classificateur (en termes de précision), l'ajout de neurones supplémentaires dans les couches cachées permet généralement de résoudre le problème. En outre, la portée originale de cette méthode est d'introduire un nouvel outil d'explication pour la classification de texte basée sur un réseau neuronal (profond). Bien qu'une précision élevée soit nécessaire pour produire des marqueurs linguistiques significatifs, le développement d'une nouvelle architecture plus performante que l'état de l'art en matière de classification de textes sort du cadre de cette étude.

4.7.2. Évaluation

Afin de comparer les caractéristiques extraites par chaque méthode (**TDS**, couche d'attention, LIME-CNN, LIME-RNN, T-CAM-CNN, T-CAM-RNN), nous considérons trois sous-ensembles de données de test présentés dans la section précédente. Plus précisément, pour chaque ensemble de données, 1 000 échantillons de test sont choisis uniformément au hasard parmi les échantillons assignés à la même classe par les architectures CNN et RNN. Les six méthodes d'explication comparées sont calculées à partir des mêmes architectures : i) **TDS**, LIME-CNN et T-CAM-CNN de l'architecture CNN de la figure 4.5 et ii) la couche d'attention, LIME-RNN et T-CAM-CNN forment le RNN de la figure 4.6.

Pour chaque échantillon, la méthode explicative produit un vecteur (*score vector*) dont la longueur est égale au nombre de tokens de l'échantillon. La m-ième entrée du vecteur est le score d'activation pour la classe prédite, attribuée au m-ième token par la méthode correspondante. Intuitivement, plus ce score est élevé, plus le token a contribué à la classification. Par souci d'exhaustivité, nous rappelons que pour chaque échantillon, LIME et T-CAM produisent autant de vecteurs que le nombre de classes. Cependant, afin d'être équitables par rapport au **TDS** et à la couche d'attention, nous ne considérons que les vecteurs de score pour la classe prédite.

La comparaison des vecteurs de score n'est pas simple, puisque chaque méthode produit des scores variant dans des plages différentes. Cependant, ce qui nous intéresse vraiment est la hiérarchie attribuée par chaque méthode aux tokens d'entrée, nous avons donc opté



(a) ACL-IMDB. (b) AG-News. (c) C50.

FIGURE 4.8 – Projection ACP des distances moyennes pour les trois ensembles de données.

pour la normalisation standard suivante :

$$w_i = (v_i - \overline{v})/\sigma_v,$$

où v_i désigne ici la ième entrée d'un vecteur générique v, $\overline{v}$ est la valeur moyenne sur les entrées, σ_v est l'écart type et w_i est la normalisation de v_i.

Après normalisation, une matrice de distance (carrée) à 6 lignes a été construite pour chaque ensemble de données, dont l'entrée (i, j) est la distance euclidienne moyenne entre les vecteurs de score pour les méthodes explicatives i et j, respectivement. La moyenne a été calculée sur les 1 000 points (échantillons) de données d'entrée pour chaque ensemble de données.

La figure 4.7 montre un clustering hiérarchique obtenus à partir des matrices de distance, pour chaque ensemble de données. Comme on peut le voir, dans tous les ensembles de données, T-CAM est groupé avec LIME avant **TDS** et l'Attention, indépendamment de l'architecture (CNN ou RNN). Il est intéressant de noter que dans le cas de la classification binaire, LIME-RNN est encore plus similaire à T-CAM-RNN qu'à LIME-CNN.

Ces résultats suggèrent une meilleure similarité entre les caractéristiques linguistiques clés détectées par T-CAM et LIME, par rapport aux marqueurs obtenus avec **TDS** ou l'Attention.

Ces résultats peuvent également être analysés en utilisant différentes approches telle que l'analyse en composantes principales (ACP). La figure 4.8 montre les deux premiers facteurs d'une ACP sur la matrice de distance euclidienne, confirmant les résultats obtenus avec le clustering hiérarchique.

La section suivante 4.8 rapporte une expérience différente pour mettre l'accent sur la plus-value linguistique de cette approche. Un jeu de données réelles impliquant une tâche de classification multi-classes est considéré afin d'évaluer la pertinence des marqueurs détectés par T-CAM.

4.8. T-CAM et linguistique

Chaque auteur possède une identité discursive faite de choix lexicaux et grammaticaux identifiables. L'analyse du discours politique est l'un des défis majeurs de la linguistique dans l'analyse des données textuelles. L'apprentissage profond apparaît comme une nouvelle façon d'interpréter les discours. Nous allons maintenant voir comment T-CAM permet aux linguistes d'effectuer une analyse critique du discours politique.

Nous nous concentrons sur le discours inaugural de Joe Biden de 2021, le 47ème président des Etats-Unis d'Amérique. Pour effectuer une analyse linguistique de ce discours basée sur les marqueurs de T-CAM, nous nous sommes appuyés sur un jeu de données composé de 1,8 million de mots issus des discours officiels des présidents américains, de J.F. Kennedy (1961) à D.J. Trump (2020). Nous avons entrainé le modèle CNN décrit dans la section 4.7.1 pour prédire le président qui a prononcé chacun de ces discours. L'ensemble des données a été divisé en 36 000 échantillons de 50 mots et nous avons conservé 80% des données comme ensemble d'entrainement, 10% pour la validation et 10% pour le test du modèle. La précision finale du modèle formé est de 84,06% sur l'ensemble de données de test. Notons que le modèle n'a pas été entrainé sur les discours du président Biden. En effet, l'idée de l'expérience est de décrire l'identité discursive du nouveau président comme une combinaison des profils linguistiques de ses prédécesseurs, en se basant sur les probabilités d'attribution du discours inaugural à chaque classe (voir *intertexte* - chapitre 7). Nous avons donc divisé le discours inaugural en échantillons de 50 mots et avons passé chaque échantillon au modèle pour une prédiction.

Comme pour de nombreuses approches statistiques (Baker 2010 ; Gisle 2020), l'apprentissage profond tient compte principalement des variations linguistiques dépendantes de la chronologie. Le corpus présidentiel américain couvre une large période de temps (presque 60 ans) : les guerres, les crises économiques, les évolutions technologiques et maintenant la crise pandémique ont influencé le discours politique au fil du temps. Le discours d'investiture de Joe Biden est un bon exemple de la façon dont le temps et une crise majeure peut affecter un discours politique, en effet notre modèle a attribué 40% des échantillons à D. Trump, principalement en raison d'un vocabulaire partagé orienté vers la crise sanitaire qui s'est répandue dans le monde entier depuis 2019.

T-CAM introduit un niveau supérieur de description, en utilisant des marqueurs discriminants pour chaque classe, il est possible d'avoir plusieurs interprétations d'un même paragraphe. Pour illustrer cette plus-value descriptive, voici un extrait tiré du discours d'investiture de Joe Biden, qui a été principalement attribué à D.J. Trump par notre CNN :

> *[...] good jobs . We can teach our children in safe schools . We can overcome this deadly virus. We can reward work, rebuild the middle class, and make health care secure for all. We can deliver racial justice. We [...]*

Dans cette séquence, T-CAM révèle que la classe "D. Trump" a le score d'activation le plus élevé sur les mots *deadly virus* (en bleu). C'est la raison principale pour laquelle cet échantillon est prédit comme "D. Trump". Cependant, si l'on considère que ce vocabu-

laire est plus spécifique au contexte de la pandémie qu'au président, ce marqueur n'est pas vraiment pertinent pour interpréter le discours de Joe Biden. Heureusement, T-CAM nous permet d'explorer différentes voies interprétatives avec différents marqueurs linguistiques liés à d'autres présidents/classes. Lorsque nous examinons la classe "Obama" par exemple, nous constatons une augmentation de l'activation des mots *the middle class* et *santé* (en orange), figure 4.9.

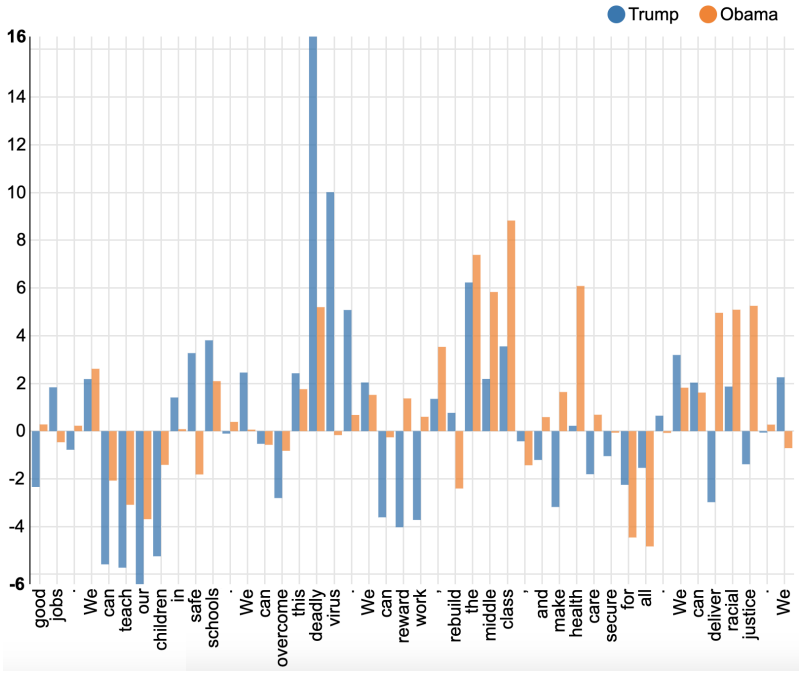


FIGURE 4.9 – Extrait du discours d'investiture de Joe Biden mis en évidence avec T-CAM par rapport aux deux présidents "Obama" et "Trump".

Ces marqueurs linguistiques sont plus pertinents en termes d'analyse du discours politique et plus proches de la position politique réelle d'un président. Les préoccupations sociales sont en fait des fils conducteurs partagés par les derniers démocrates (B. Obama et B. Clinton). Cette observation est rapidement confirmée par un pur argument statistique de fréquence (figure 4.10).

Il faut souligner ici que l'histogramme de la figure 4.9 est conceptuellement équivalent aux sorties machines sur la classification d'images fournies par l'algorithme d'origine CAM (ZHOU et al. 2016). En effet, comme dans ce travail, CAM met en évidence les régions pertinentes de l'image d'entrée, utilisées par le classificateur pour l'attribution des classes, ici l'histogramme de la Figure 4.9 met en évidence les mots pertinents du texte d'entrée activant chaque classe.

De plus, l'histogramme peut être généralisé à toutes les classes (anciens présidents) et appliqué à l'ensemble du discours inaugural de Joe Biden (tous les échantillons). Dans ce

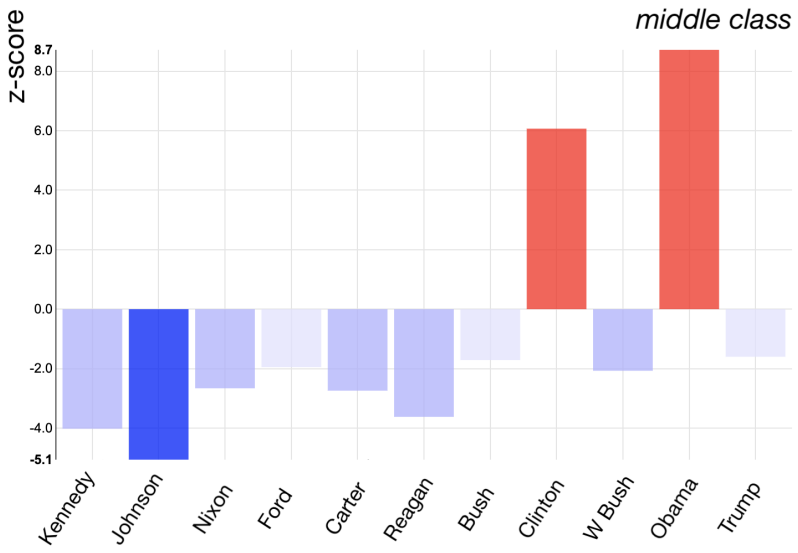


FIGURE 4.10 – z-score du bi-gramme *middle class* pour les présidents américains. Plus le score z est élevé, plus le bigramme est surutilisé (et vice versa).

cas, le T-CAM peut être utilisé pour interpréter le discours en observant l'intertextualité entre J. Biden et ses prédécesseurs. Cette analyse sur le discours inaugural peut être résumé ici avec une courte liste des motifs les plus forts (taux d'activation) que donne T-CAM pour chaque classe (Tableau 4.1).

Trump	Obama	W.Bush	Clinton
much to	*humility*	*violence*	*health care*
a raging virus	*middle class*	*will defend*	*racism*
once-in-a	*challenges*	*a culture*	*our children*
century virus			

TABLE 4.1 – Les motifs du discours du président Biden, ayant le plus haut score d'activation T-CAM pour chaque classe.

T-CAM montre donc qu'il est possible d'ajuster le score de chaque mot en utilisant les dernières couches du réseau (*Denses*). Nous alons voir maintenant comment cette méthode permet aussi de corriger le **TDS** et de lui donner une valeur interprétative supplémentaire.

4.9. TDS pondéré

Les couches de sortie du réseau (*Dense*) ont donc pour rôle d'encoder et de compresser l'information apprise dans les couches précédentes (l'embedding et la convolution) pour obtenir en sortie un seul vecteur correspondant à la prédiction du modèle. Dans cette couche, le **TDS** calculé en Eq. (3.1) peut aussi être modifié, accentué et même potentiellement inversé en utilisant l'agorithme de T-CAM. Le but du modèle est en effet d'activer la bonne classe, le bon neurone correspondant, dans la couche de sortie, un vecteur que

nous appellerons $yi \in \mathbb{R}^K$ où K représente le nombre de classes à prédire.

Pour obtenir une version pondérée du calcul du **TDS** il faut donc remplacer la somme Eq. (3.1) par un calcul matriciel plus complexe qui prend en compte toutes les couches du réseau jusqu'à la sortie. Dans la section 3.4 nous avons vu que la *déconvolution* nous donne un ensemble de vecteurs $x_{i1}, \ldots, x_{iM}$. Pour simplifier le calcul, ces vecteurs peuvent être concaténés en un seul vecteur colonne X_i de taille $D\text{x}M$, la *déconvolution* s'écrit alors :

$$y_i = d + C\left(\varphi\left(b + AX_i\right)\right) \tag{4.3}$$

Où $A \in \mathbb{R}^{E \times DM}$, $b \in \mathbb{R}^E$, $C \in \mathbb{R}^{E \times K}$ et $d \in \mathbb{R}^K$. E est la taille de l'avant-dernière couche (les filtres) de la *déconvolution* et φ est la fonction d'activation de la partie *dense* du réseau. Pour obtenir un score spécifique à chaque token d_{im} on observe donc que :

$$AX_i = \sum_{m=1}^{M} A_m x_{im}^T \tag{4.4}$$

Où $A_m \in \mathbb{R}^{E \times D}$ est la sous matrice A obtenue en sélectionnant les D colonnes depuis la $(D(m-1)+1)$-ième jusqu'à la $(D(m-1)+D)$-ième. Nous pouvons définir ainsi la formule pondérée du **TDS** :

$$wTDS(d_{im}) := d + C\left(\varphi\left(b + A_m x_{im}^T\right)\right) \tag{4.5}$$

Remarquons que contrairement à $TDS(d_{im})$, $wTDS(d_{im})$ est maintenant un vecteur de K dimensions. Chaque dimension représente l'activation du mot d_{im} (conditionné au contexte) pour la classe K. De plus, la multiplication de la matrice $A_m x_{im}$ induit donc la pondération de la somme des éléments x_{im} qui manquait à la simple somme définie par le **TDS** Eq. (3.1). Pour cette raison nous appelons cette nouvelle mesure Eq. (4.5) le **weighted Text Deconvolution Saliency (wTDS)**.

Avec ce nouveau calcul, le **wTDS** montre un aspect différent de la convolution. Le **wTDS** ne donne plus simplement pour chaque mot une valeur unique, mais bien maintenant autant de valeurs qu'il y a de classes dans le modèle. Il existe donc maintenant plusieurs lectures possibles de la convolution, une pour chaque classe, qui sont regroupées dans le vecteur calculé par le **wTDS**.

Reprennons alors l'exemple de la section 3.5 qui illustre les limites du **TDS**.

> *[...] neighborhoods for their families , and good jobs for themselves . These are just and reasonable demands of righteous people and a righteous public. But for too many of our citizens , a different reality exists : Mothers and children trapped in poverty in our inner cities ; rusted-out [...]*

Avec le **wTDS**, la convolution permet de projeter plusieurs sorties possibles pour l'extrait de D. Trump qui rendent cohérentes les observations du *z-score*, de LIME et du **wTDS**, figure 4.12.

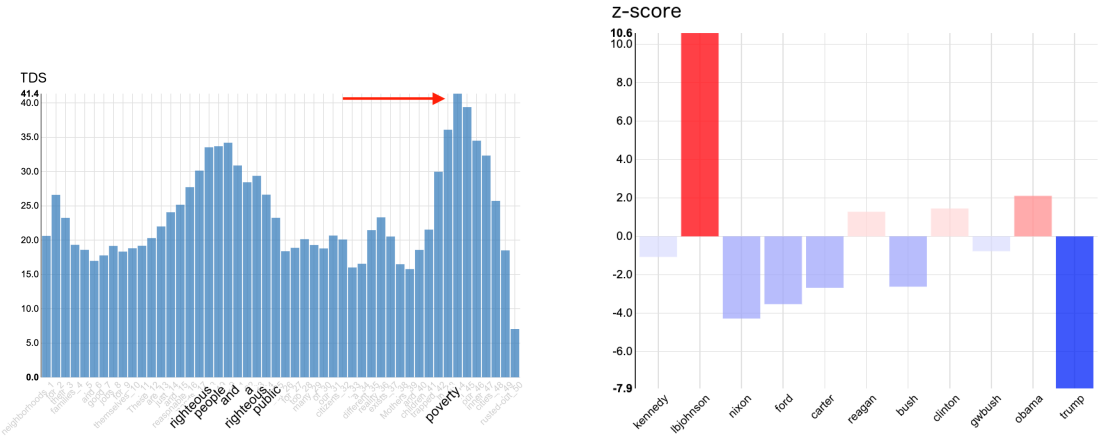


FIGURE 4.11 – Comparaison du **TDS** et du z-score de *poverty* dans le discours présidentiel américain

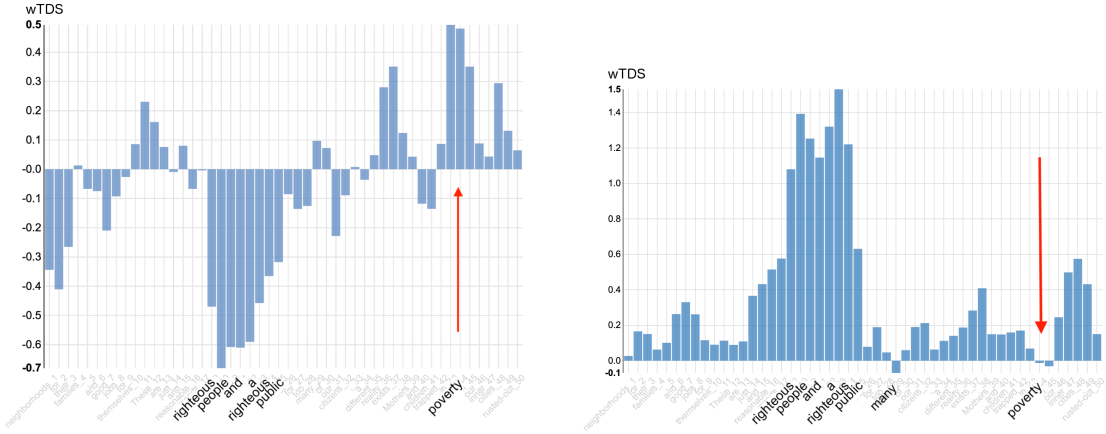


FIGURE 4.12 – **wTDS** pour les classes L. B. Johnson à gauche et D. Trump à droite

L'analyse de l'extrait du discours de D. Trump devient plus évidente à interpréter maintenant. C'est le passage populaire autour de *righteous*, *people*, *public* qui reste très activé et qui pousse visiblement le réseau à attribuer cette séquence de mots au président américain. Le mot *poverty* dans son contexte est quant à lui maintenant invisible, voire négatif comme pressenti. À noter que d'autres phénomènes apparaissent maintenant de manière plus pertinente, comme la prise en compte du contexte du mot *many* qui ne semble pas avoir contribué à l'attribution de l'extrait. Certes *many* comme beaucoup de superlatifs est très spécifiques statistiquement de D. Trump (z-score de +10.0) qui aime utiliser un discours hyperbolique. Mais ici, le président en fait un usage qui lui ressemble visiblement moins. En effet, il utilise le mot *many* précédé de l'adverbe *too*. Or la spécificité du segment *too many* est négative pour Trump (*z-score* de -2.1 : figure 4.13). Il est dès lors logique que le **wTDS** n'affiche pas d'activation positive de ce mot dans ce contexte.

La plus-value descriptive de la convolution semble ainsi attestée. En étant capable d'in-

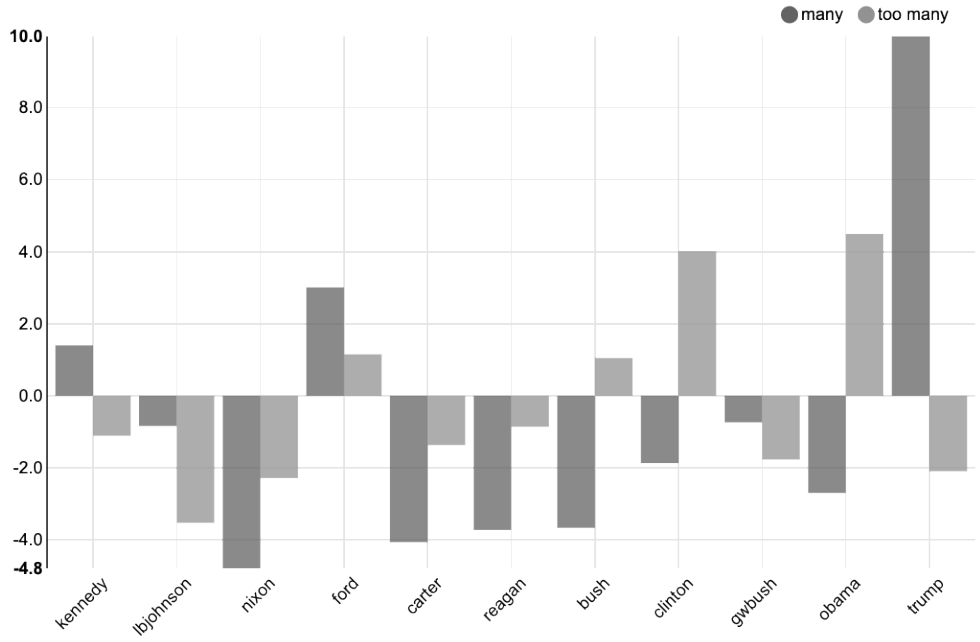


FIGURE 4.13 – Comparaison du z-score de *many* et *too many* dans le discours présidentiel américain

terpréter dans un segment chaque mot dans son contexte, le *deep learning* associe les calculs statistiques connus (*z-score*, cooccurrence, segment répété ...) avec une approche séquentielle ou segmentale du texte pour en extraire, avec le calcul du **wTDS**, l'ensemble des particularités linguistiques différentes pour chaque classe. La statistique descriptive classique est donc complétée par un parcours interprétatif d'un genre nouveau qui s'ajoute aux outils traditionnels de l'ADT. La version pondérée du **TDS** permise par l'algorithme de T-CAM semble être la réponse à la description des données par le *deep learning*. La chaîne d'activation de neurones qui part de l'*embedding* des mots jusqu'à la couche de sortie du réseau révèle ainsi l'information cachée du réseau. Cette information, qualifiée souvent de boite noire, est en réalité tout à fait interprétable par l'humain. Sur le principe de l'analyse de contraste d'une image, il est possible avec le **wTDS** de rendre compte maintenant des contrastes des textes.

Il reste quelques restrictions à l'usage du deep learning par rapport à l'ADT classique. Contrairement à la statistique, le deep learning ne peut décrire un corpus que par l'usage de nouveaux textes inconnus de l'ensemble d'entrainement. Une prédiction sur un texte présent pendant l'apprentissage n'a aucun sens. La probabilité en sortie sera proche de 100% pour l'ensemble des segments qui composent ce texte et il sera alors difficile d'extraire un passage significatif à interpréter. Décrire un corpus par deep learning nécessite donc de commencer par le retrait d'un ensemble de textes pour pouvoir les soumettre ensuite à la tâche de prédiction.

De plus, l'algorithme d'apprentissage qui boucle plusieurs fois sur l'ensemble des données peut rendre les réseaux de neurones trop sensibles aux basses fréquences, c'est-à-dire

aux mots ou expressions rares, uniquement présentes dans un texte et qui deviennent des caractéristiques suffisantes pour identifier les textes. De simples noms propres ou des dates peuvent rapidement devenir les seules saillances détectées par le deep learning qui passe alors à côté d'objets linguistiques plus complexes et potentiellement plus pertinents pour l'analyse descriptive attendue par la méthode.

Chapitre 5

Corpus et passages-clés

Ce chapitre reprend pour l'essentiel le chapitre d'ouvrage Vanni *et* Precioso *2021 ainsi que l'article* Vanni, Corneli, *D.* Longrée *et al. 2020b. Des modificiations mineures et certains enrichissements ont été apportés pour structurer la présentation des différents résultats.*

5.1. Introduction

Nous avons vu que les couches cachées du deep learning pouvaient être interprétables par l'humain. Chaque partie du réseau peut être analysée séparément ou au contraire intégrée dans le traitement global du modèle. Nous avons là un moyen de comprendre et de nous servir de l'intelligence artificielle d'une nouvelle façon. La boite noire laisse place à des algorithmes descriptifs avancés qui rendent compte des choix du modèle.

Néanmoins la linguistique textuelle doit maintenant intégrer ces résultats et proposer une articulation forte entre les outils statistiques classiques et les réseaux de neurones profonds pour pouvoir améliorer les qualités descriptives de notre modèle. D'une part les textes doivent être contrôlés en amont de l'apprentissage par la statistique et d'autre part l'ADT se doit de contextualiser tous les résultats descriptifs du deep learnig avec nos outils traditionnels.

Un des concepts linguistiques forts qu'il est possible d'associer aux sorties machines du modèle est la notion de *passages* dans le corpus. La définition que donne (Rastier 2007) semble faire écho à ce qui est observé avec le **wTDS** : un ensemble de zones d'activation plus ou moins longues sur lequel porte notre interprétation. Dans ce cadre théorique, l'ADT tente depuis des décennies d'extraire automatiquement des passages clés (Vanni, Mayaffre et D. Longrée 2018) : ce sont les passages interprétables, les plus discriminants statistiquement pour le corpus et le partitionnement choisis. Avec le deep learning et le modèle convolutionnel proposé, les passages-clés deviennent les passages du corpus identifiés par le **wTDS** qui activent le plus fortement chaque classe en sortie de réseau. Pour parvenir à les observer avec notre modèle, la réutilisation des données d'entrainement est alors nécessaire.

5.2. Passages-clés bruts

Dans l'architecture proposée, le corpus est fragmenté en morceau de taille fixe pour pouvoir être analysé par la convolution qui considère le texte comme une donnée spatiale et statique. Chaque morceau, fenêtre ou segment sont utilisés pour superviser l'entrainement. Le poids des neurones est corrigé à chaque itération jusqu'à faire correspondre un maximum de segments aux bonnes classes. Au fil de l'apprentissage, le modèle peut converger vers un état où l'ensemble des segments est correctement attribué, l'*accuracy* de l'ensemble d'entrainement atteint alors 100%, c'est le surapprentissage. Mais qu'il y ait surapprentissage ou non, l'algorithme de descente de gradient pousse toujours le système à maximiser en sortie la valeur de la probabilité qui atteint alors 1 pour les segments les plus forts.

Cette normalisation des données semble incompatible avec la détection des passages-clés dans un corpus qui nécessite de conserver une trace de la hiérarchie entre ces segments. Or cette hiérarchie existe toujours dans les données d'apprentissage, elle est simplement masquée par la dernière fonction d'activation du réseau, la fonction qui génère justement la probabilité de sortie.

Soit le nombre de classes du modèle K, nous pouvons noter $x_i \in \mathbb{R}^K$ la valeur de la couche de sortie du i-ième texte à classer. Donc x_{ik} est la valeur du k-ième neurone qui est un nombre réel qui correspond à la k-ième classe du modèle. La fonction d'activation *softmax* est alors en général appliquée à x_i pour obtenir K probabilités p_{ik} :

$$p_{ik} = \frac{\exp(x_{ik})}{\sum_{j=1}^{K} \exp(x_{ij})}. \tag{5.1}$$

La plus haute probabilité observée correspond ainsi à la classe attribuée. Cependant, si la valeur de sortie d'un des neurones de x_i est significativement supérieure aux autres elle sera arrondie à 1 par la transformation *softmax* et tous les autres neurones à zéro. C'est précisément cette perte d'information qui rend impossible le tri des segments de textes utilisés pendant l'apprentissage. Considérons deux segments de texte correspondant à deux sorties $x_i \neq x_j$ avec des activations suffisamment élevées, mais correspondant tous deux à la même k-ième classe du modèle, l'application du *softmax* rendra alors le classement hiérarchique impossible, les deux valeurs du k-ième neurone seront égales à 1.

Pour contourner ce problème, l'astuce est d'observer la sortie du réseau juste avant cette dernière fonction d'activation. Ainsi, la valeur de sortie correspond simplement à la somme des activations de la couche précédente (pondérée par les poids des neurones de la dernière couche) et aucun ordre hiérarchique n'est alors perdu entre les segments analysés (figure 5.1). Le vecteur x_i ne correspond alors plus à un ensemble de probabilités, mais à un vecteur de nombres réels dont la plus grande valeur correspond à la classe attribuée. Cette valeur n'étant plus lissée par le *softmax*, chaque niveau d'activation pour chaque classe peut alors être comparé entre les différents segments de texte du corpus.

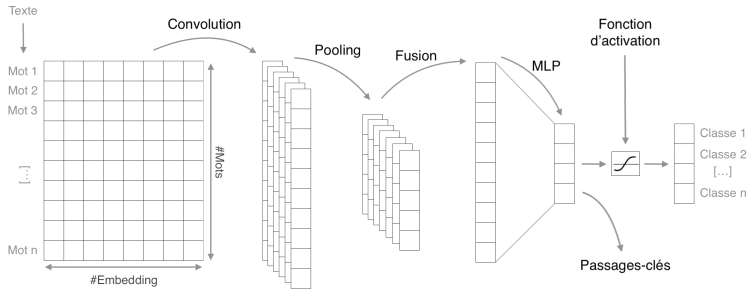


FIGURE 5.1 – Détection des passage-clés

Cette méthode permet donc de réutiliser l'ensemble des données d'entrainement lors d'une tâche de prédiction. Il s'agit là d'une utilisation détournée de la fonction de prédiction du deep learning justifiée par le fait que l'architecture cherche à convertir les performances prédictives des réseaux de neurones en performances descriptives pour l'ADT. Comme attendu, il existe une véritable mine d'informations dans les segments d'apprentissage du corpus. Le tri des différentes séquences avec cette méthode permet d'observer les passages identifiés par le **wTDS** dans un contexte où les classes activées en sortie sont hiérarchiquement très fortes. Ces passages repérés alors automatiquement sont ainsi appelés les **passages-clés** du deep learning.

Pour illustrer cette détection automatique des passages-clés nous utilisons le corpus français qui regroupe l'ensemble des discours officiels des présidents français de de Gaulle à Macron, soit environ 2.7 millions de mots pour 8 locuteurs à reconnaître. Bien que ce corpus soit amené à grossir au fil des nouveaux discours et des nouveaux président, il peut être utilisé à un instant donné pour décrire les motifs caractéristiques de chaque président (en tant que corpus exhaustif donc). Par exemple l'analyse du discours de Macron (c.-à-d. l'ensemble de ses discours) donne des résultats stupéfiants qui mettent en valeurs les marqueurs linguistiques les plus caractéristiques du discours de l'actuel président. À titre d'illustration le segment qui arrive en tête du classement regroupe à lui seul un ensemble de passages (soulignés par le **wTDS**) très significatifs :

> *[...] choix en brutalisant les consciences , des convictions profondes que je respecte chez chacune et chacun . Et donc je souhaite que durant l' année 2018 nous puissions avoir ce débat de manière apaisée . Nous aurons à revenir sur les lois de bioéthique en 2018 . Le comité d' [...]*

(Extrait de l'interview télévisé de TF1 du 15 août 2017)

Il y a tout d'abord le passage qui commence avec le syntagme *Et donc* qui est typiquement macronien. Avec un indice de spécificité de +21.7 cette expression permet à l'actuel président de sur-marquer une causalité (le « et donc » est fautif grammaticalement) à tout argument qu'il emploie dans son discours. Suivi du mot *souhaite*, une autre spécificité forte (+9.5) nous avons là un enchainement particulièrement caractéristique d'E. Macron. La deuxième zone forte du **wTDS** est une séquence de mot autour d'un verbe conjugué au subjonctif à la 1ère personne du pluriel (*puissions*) qui introduit le mot *débat*

forte spécificité marcroniene. Si le subjonctif ou la forme graphique *puissions* ne sont pas en eux-mêmes spécifiques d'E Macron, l'analyse des cooccurrences montre que *débat* est souvent associé dans le discours du président, à cette modalité verbale. Pour 808 *débat* prononcés dans l'ensemble des discours, une douzaine sont dans l'environnement particulier de *puissions* ce qui donne un indice de cooccurrence significatif de $+4,8$. Ces passages, unités phraséologiques plus ou moins construites permettent, lorsqu'ils sont concentrés dans une même fenêtre, de définir des passages-clés, et de discriminer fortement l'actuel président de ses prédécesseurs.

Néanmoins il semble que certaines formes graphiques soient encore fragiles quant à leur plus-value heuristique sur notre corpus. C'est le cas de l'année *2018* qui est soulignée à plusieurs reprises dans les différents segments de textes. Cette date est prononcée principalement par Macron dans le corpus pour des raisons chronologiques évidentes, et n'a dès lors peu de valeur interprétative pour l'historien ou le linguiste. Si nous regardons plus loin dans les autres segments, ce sentiment est confirmé par l'apparition d'autres dates ou encore des noms propres (« Castaner », « Philippe ») particuliers au quinquennat du président. Ces éléments que la statistique repère généralement peuvent être considérés comme des artefactes et constituer un frein à la pertinence descriptive du *deep learning*. Nous allons voir maintenant qu'il est possible d'utiliser en amont du *deep learning* les méthodes et outils de l'ADT standards pour contourner ce problème.

5.3. Ajustement du corpus

Il est fréquent en statistique textuelle de surveiller les déséquilibres qui existent dans un ensemble de données. L'interprétation des résultats est dépendant de l'homogénéité des données et l'ADT fait rarement l'impasse sur le nettoyage du corpus. Comme le regrette François Rastier, pour le deep learning, les volumes de données nécessaires à l'apprentissage semblent en général prépondérer aux choix des données (Rastier 2021). Toutefois, l'apparition d'études récentes comme les exemples adversaires (Yuan et al. 2019) en images, les IA conversationnelles racistes ou sexistes (Wiegand, Ruppenhofer et Kleinbauer 2019) en génération de texte ou encore les systèmes critiques (voitures autonomes, médecine, ...) poussent la communauté à revoir ses priorités.

Nous avons vu que le calcul des passages-clés et l'analyse du **wTDS** dans un corpus demandent une attention particulière au corpus. A l'usage, les éléments principaux qui discriminent un texte sont souvent hélas associés à un caractère mal encodé ou une typographie systématiquement défaillante, plus qu'à des choix sémantiques ou syntaxiques de l'auteur. Si en statistique il est facile simplement d'ignorer un mot dans une AFC ou dans une liste de spécificités, les algorithmes de classification automatique sont tous impuissants devant ces biais et encore plus lorsqu'il s'agit d'apprentissage supervisé comme en *deep learning*.

La solution pour contourner ce problème vient, comme en ADT, du prétraitement des données. Cette sélection préliminaire des données peut soit être manuelle avec une bonne connaissance du corpus, soit semi-supervisée en utilisant certains résultats statistiques, soit totalement automatique basée sur des règles statistiques précises. Le nettoyage manuel

étant de l'ordre de l'expertise humaine, nous allons nous concentrer sur les deux autres cas et voir leur impact sur la qualité des résultats obtenus.

5.3.1. Traitement hybride

Le calcul des spécificités est un excellent moyen d'avoir un aperçu des biais les plus flagrants dans un corpus. Le calcul généralisé sur l'ensemble des métadonnées disponibles permet ainsi d'obtenir une liste dont les premiers éléments sont par nature les plus discriminants. Il est alors facile ensuite de retirer du corpus d'entrainement les mots spécifiques pour lesquels il n'y a pas d'interprétation autrement qu'évidente. Un autre moyen est d'utiliser un calcul de distance intertextuelle (C. Labbé et D. Labbé 2003) ou une analyse factorielle sur les mots les plus fréquents pour apprécier la justesse et l'équilibre d'un corpus.

L'idée est donc d'articuler la connaissance statistique des données avec le traitement *deep learning*. L'application du modèle doit se faire dans le cadre de ce prétraitement et les biais identifiés doivent être supprimés ou rendus négligeables dans l'ensemble des données d'apprentissage. C'est pourquoi l'implémentation de cette architecture est totalement intégrée à Hyperbase Web, la plateforme de référence ici pour l'ADT, qui permet de préparer un corpus et de sélectionner directement dans l'interface la liste des mots et des biais à supprimer lors de l'apprentissage (voir chapitre 8). Mais qu'elle soit intégrée ou non à une plateforme d'ADT, cette architecture propose aussi une manière non supervisée de préparer les données.

5.3.2. Traitement automatique

Nous avons vu que le calcul des spécificités en ADT est un moyen efficace d'apercevoir les traits saillants d'un corpus. À partir de quatre valeurs simples k, f, t, T, le calcul historique du *z-score* s'écrit :

$$z = \frac{k - fp}{\sqrt[2]{fqp}}. \tag{5.2}$$

Où k est la fréquence du mot observée dans le texte, f la fréquence du mot observée dans le corpus, t l'étendue du texte, T l'étendue du corpus et avec avec $p = \frac{t}{T}$ et $q = 1 - p$. Notons que ce calcul a laissé place à un autre calcul plus exact depuis (Lafon 1980). Le calcul de la probabilité pour un mot d'avoir la fréquence k dans un texte, fondé sur la distribution hypergéométrique, s'écrit ainsi :

$$p(x = k) = \frac{f!(T-f)!t!(T-t)!}{k!(f-k)!(t-k)!(T-f+t+k)!T!}. \tag{5.3}$$

Il s'agit donc de mesurer la distribution d'un mot dans le corpus en nous rapportant à un écart par rapport à la valeur théorique attendue dans le corpus. En utilisant les spécificités, certaines règles peuvent ainsi être appliquées de manière automatique pour contrôler

le poids des spécificités trop fortes et ainsi retirer les tokens les plus discriminants. L'idée est de forcer le *deep learning* à apprendre de nouvelles stratégies de classification qui reposent sur des marqueurs linguistiques plus complexes et moins évidents que la spécificité pour donner un regard nouveau sur le corpus. Si nous devions comparer cette approche avec une classification d'images, nous pourrions dire que nous souhaitons que la machine soit capable de classer les chiens et les chats tout en la privant d'informations certes utiles mais grossières telles que les oreilles, les moustaches et la queue. C'est ce qu'il est possible de faire sur le texte avec ces quelques règles qui suppriment l'information trop attendue.

Soit un seuil α tel que α tel que $\alpha = k \times \epsilon$ avec $\epsilon = k \times 0.1$ au-delà duquel la fréquence d'un mot observé dans le corpus f devient suffisamment élevée pour être considéré par le modèle. Tous les mots tels que $f < \alpha$ sont éliminés et deviennent négligeables pour le deep learning. Ce procédé permet de supprimer une partie des mots exclusifs ou quasi-exclusifs d'une partie du corpus, tout en conservant les spécificités fortes et suffisamment fréquentes.

Les autres règles appliquées dépendent de la nature du corpus étudié. Dans les corpus diachroniques comme celui des présidents français, il convient de supprimer par exemple l'ensemble des dates et autres chiffres qui dévoilent souvent de manière trop évidente l'articulation chronologique des données d'entrainement. Il est possible aussi d'enlever la ponctuation trop sensible aux effets de retransmissions des textes (choix arbitraire des éditeurs successifs à l'Elysée).

Un des effets attendus de ce filtrage est la diminution des scores d'*accuracy*. Mais ce prétraitement apparait nécessaire, car il augmente considérablement la plus-value heuristique des sorties machines descriptives du *deep learning*. Encore une fois, l'expertise du corpus et l'analyse statistique sont les maîtres mots d'une analyse *deep learning* de qualité pour la linguistique des textes.

5.4. Passages-clés filtrés

Pour illustrer la plus-value de ces règles de filtrage, l'exemple suivant reprend et compare celui de la section 5.2 au nouveau calcul des passages-clés qui prend une dimension descriptive bien plus intéressante :

> *[...] aussi de compenser les profondes désorganisations sur l' économie traditionnelle que cette transformation parfois crée . Les grandes plateformes numériques , la protection des données sont au cœur de notre souveraineté à cet égard . Et il en est de même pour la taxation , nous devons avoir ce débat [...]*

(Discours d'E. Macron sur l'Europe à Paris, le 26 septembre 2017)

Dans cet extrait les passages-clés affichés par le **wTDS** montrent des thématiques et des choix politiques forts de l'actuel président. Aucune date ou biais particuliers ne viennent plus polluer l'interprétation, et l'essentiel du discours d'E. Macron est alors retrouvé. L'idée principale du *débat* est toujours présente, associée au verbe *devoir* conjugué au présent et à la première personne du pluriel avec le pronom personnel *nous* qu'affectionne tout particulièrement le président. A noter aussi le mot *transformation*, une spécificité

partagée avec de Gaulle qu'il utilise pour souligner le mouvement et le caractère inédit de son mandat, le « nouveau monde » comme il l'appelle, avec des thématiques fortes comme le *numérique* avec lequel il souhaite redéfinir notre *souveraineté*. Un ensemble de marqueurs significatifs donc, rassemblés dans plusieurs passages-clés évalués par le deep learning, qui montrent E. Macron comme un président En marche ou en mouvement (propositions de *débats*, *plateformes numériques*, *transformations* technologiques ; figure 5.2).

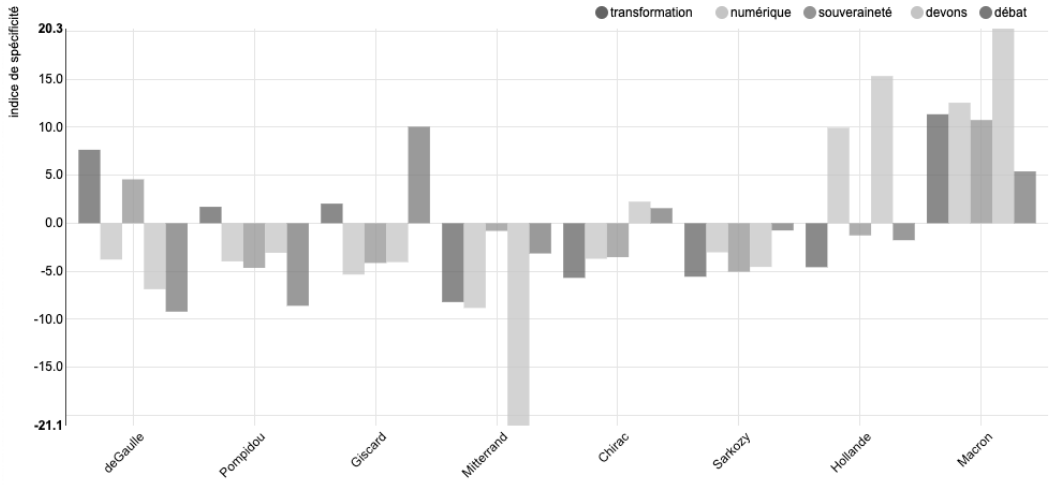


FIGURE 5.2 – Spécificités des passages-clés d'E. Macron

Ainsi, la *déconvolution* et le **wTDS** apparaissent comme des outils fondamentaux pour s'orienter vers une utilisation descriptive du *deep learning* appliqué au texte. Il convient maintenant d'insister sur la nécessetité d'étoffer l'analyse en prenant en considération différente représentation lingusitique du texte (formes graphiques, le codes gramaticaux, lemmes). En effet l'utilisation d'une seule représentation des mots (la forme graphique par défaut) en entrée du réseau touche ses limites lorsque le but de la méthode est de découvrir de nouveaux observables, potentiels *motifs* linguistiques complexes qui rendent compte de toutes les dimensions du langage. La lemmatisation et l'analyse morphosyntaxique d'un texte sont, depuis de nombreuses années, partie intégrante de l'analyse automatique ou semi automatique des textes. L'idée que nous allons explorer maintenant est donc d'ajuster notre modèle pour prendre en compte cette vision du texte qui additionne plusieurs niveaux de représentation des mots pour permettre au *deep learning* d'en restituer la plus-value descriptive attendue par la méthode.

Chapitre 6

Linguistique *multidimensionnelle*

Ce chapitre reprend pour l'essentiel le chapitre d'ouvrage Vanni *et* Precioso *2021 ainsi que l'article* Vanni, Corneli, *D.* Longrée *et al. 2020a. Des modificiations mineures et certains enrichissements ont été apportés pour structurer la présentation des différents résultats*

6.1. Introduction

Historiquement l'ADT est passée du traitement de textes bruts au traitement de textes étiquetés (Brunet 1999). Si la seule forme graphique d'un mot est souvent suffisante, il existe de nombreux cas ambigus d'homonymes où l'utilisation d'un simple dictionnaire de formes ne suffit pas. La lemmatisation est un moyen efficace de lever ces ambiguïtés et obtenir une description des mots sur plusieurs niveaux de représentation. De plus, les lemmatiseurs permettent le plus souvent d'associer au mot une étiquette morpho-syntaxique (partie du discours, temps verbal, genre, nombre). C'est cette information linguistique qui devient précieuse pour notre approche descriptive du *deep learning*, car elle permet d'observer des structures lexico-grammaticales, proches des motifs théorisés par (Mellet et D. Longrée 2009) qui associent potentiellement plusieurs niveaux de représentation du texte dans la même structure. Le modèle convolutionnel utilisé jusqu'à présent doit donc être adapté pour intégrer ce surplus d'information dans le but d'obtenir une description encore plus fine des saillances textuelles d'un corpus.

6.2. Architecture *multi-channels*

La classification d'images par réseaux convolutionnels utilise souvent plusieurs versions des images en même temps pour décomposer l'information et permettre au réseau de repérer un maximum de caractéristiques. On utilise généralement trois canaux différents, un pour chaque niveau de couleur dans le cas d'images RGB (*Red*, *Green*, *Blue*), on parle alors de réseaux *multi-channels*. Les données en entrée prennent la forme de matrices en 3 dimensions, des *tenseurs*, et chaque représentation correspond à une dimension de la matrice. La couche convolutionnelle applique ensuite des filtres sur l'ensemble de ces dimensions et fusionne les résultats, une abstraction des données *muilti-channels*, dans un seul vecteur de sortie.

Avec des textes dont les tokens ont été étiquetés, les trois niveaux de couleurs peuvent être remplacée par nos trois niveaux de représentation d'un mot : la forme graphique, la catégorie grammaticale (*part of speech* ou POS), et le lemme. L'étiquetage morpho-syntaxique a d'ailleurs déjà montré de nombreux résultats en *deep learning* comme (Collobert et al. 2011) qui tend à prouver que les modèles convolutionnels peuvent être sensibles à l'analyse de structures bien plus complexes que le simple lexique.

Chaque représentation du mot peut ainsi être vue comme une sorte d'*embedding* différent et un mot peut alors correspondre à plusieurs *embeddings* à la fois, à l'image des modèles multi-embedding comme (Ruder, Ghaffari et Breslin 2016).

Techniquement chaque *embedding* est donc ajouté en parallèle pour former un tenseur qui permet au réseau d'intégrer les trois niveaux d'information pour le même texte. Une convolution sur trois dimensions peut ainsi être appliquée pour prendre en compte cette nouvelle forme du texte. Le problème de cette méthode est qu'elle ne permet pas de conserver de traces de la participation de chaque *channel* dans la prise de décision finale. La déconvolution qui peut être appliquée ensuite ne peut simplement que reproduire le *token*, c'est à dire le mot sans distinction de sa forme, sa POS ou son lemme (figure 6.1).

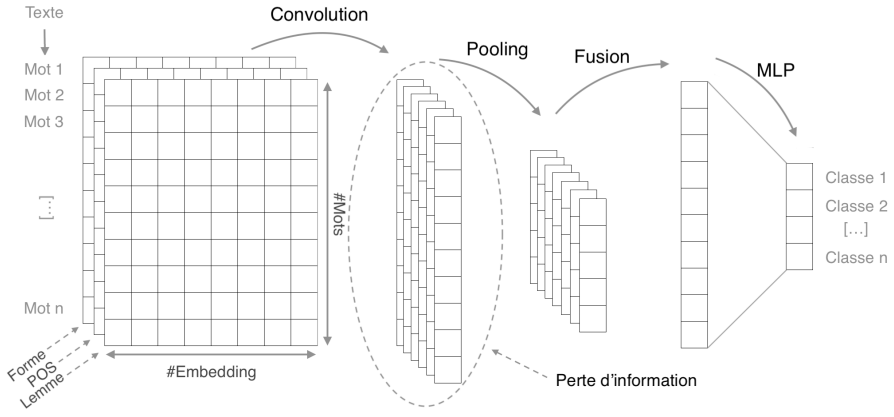


Figure 6.1 – Une convolution multi-channels 3 dimensions

Dans un souci de retour au texte permanent et pour permettre une analyse descriptive du texte sur plusieurs niveaux, il est nécessaire de modifier quelque peu l'architecture classique utilisée pour les images RGB. Pour contourner le problème de la perte d'information, il faut diviser la convolution 3D des images en trois convolutions 2D en parallèle (pour mémoire, la convolution 2D pour le texte n'agit en réalité que sur une dimension et on parle plutôt de convolution 1D, voir section 2.3.2). Cette méthode permet de fusionner l'information plus tard dans le réseau et ainsi conserver une trace de la convolution sur chaque *channel* pour pouvoir calculer le **wTDS** relatif à chaque représentation du texte. Cet algorithme s'appelle *late fusion* (figure 6.2).

Avec cette méthode, il n'y a pas une seule couche de sortie de convolution mais trois,

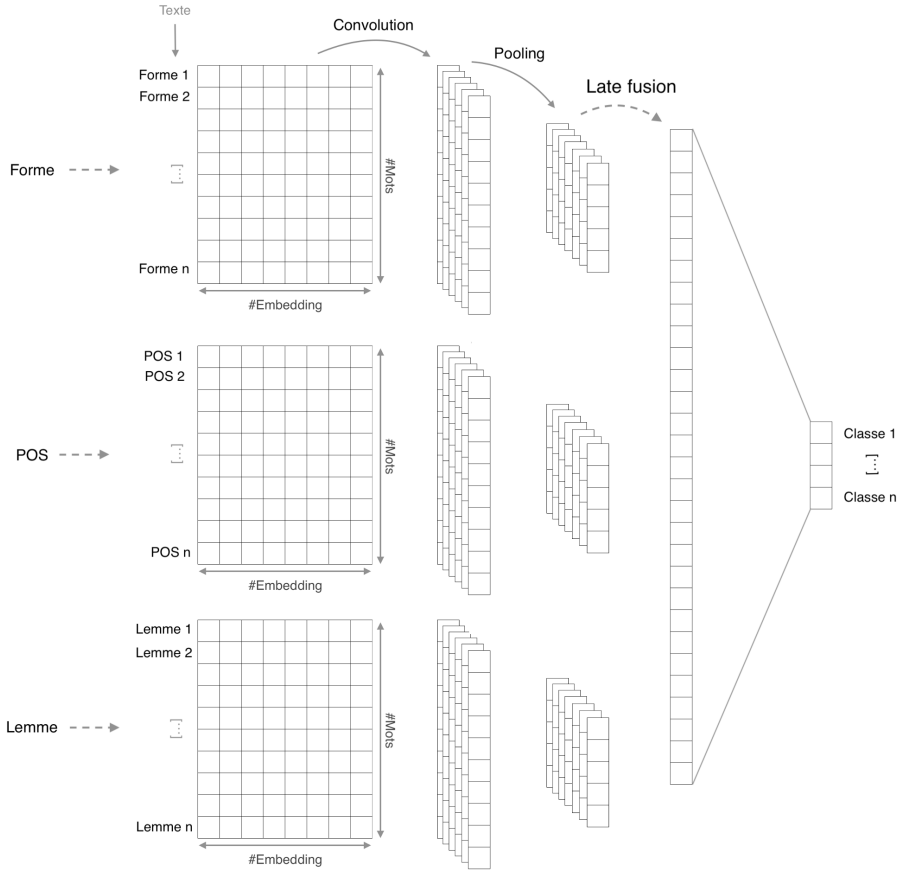


FIGURE 6.2 – Trois convolutions multi-channels 2 dimensions

une pour chaque niveau de représentation des mots. Le score de **wTDS** ainsi obtenu sur chaque channel révèle alors une information riche et non redondante illustrée par la sortie de déconvolution *multi-channels*, figure 6.3.

La sortie brute de cette triple déconvolution offre visiblement des marqueurs différents dans les trois niveaux de représentation du texte. Si on superpose les trois sorties de déconvolution, on s'aperçoit qu'ils ne correspondent pas et sont vraisemblablement complémentaires. Pour confirmer cette idée, l'information peut être hiérarchisée puis regroupée dans un sortie machine multi-niveaux qui autorise l'analyse de motifs linguistiques complexes et permet de franchir une nouvelle étape dans la description des textes par le *deep learning*.

6.3. Détection des motifs profonds

Les nouveaux observables multidimmentionnels que surligne le **wTDS** font apparaitre des caractéritiques nouvelles et profondes des textes. Ces motifs que nous appelons **motifs profonds** illustrent des analyses linguistiques souvent complexes que le modèle utilise pour pouvoir classer et reconnaitre chacun des textes issus du corpus d'entrainement. Pour illustrer ces *motifs profonds*, le prochain exemple complète celui de la section 5.2 en

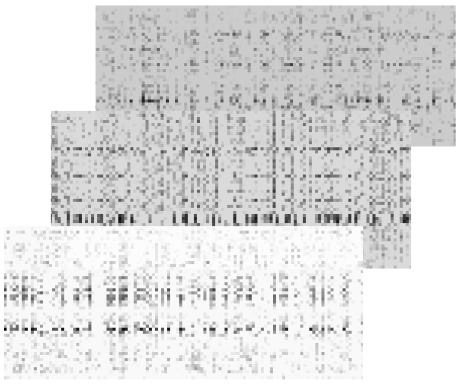


FIGURE 6.3 – Deconvolution multi-channels

proposant une nouvelle extraction de passages-clés utilisant l'architecture multi-channels et un affichage du **wTDS** sur trois niveaux (en bleu les formes graphiques, en orange les catégories grammaticales et en vert les lemmes) :

> *[...] permettre aux acteur européen d'émerger dans un marché loyal et qui VER :FUTUR aussi de compenser les profondes désorganisations sur l'économie traditionnelle que PRO :DEM transformation parfois créer. Les grandes plateformes numériques, la protection des données sont au cœur de notre souveraineté à cet égard. Et [...]*

(Discours d'E. Macron sur l'Europe à Paris, le 26 septembre 2017)

Dans cet extrait les marqueurs linguistiques repérés par le **wTDS** *multichannels* mettent en évidence des choix discursifs fort pour E. Macron. Par exemple, les verbes (lemmes) encourageants *permettre* ou *créer* sont typiques, dans un passage où le deep learning repère un verbe au futur promissif qui renforce la projection vers l'avant. Le mot *transformation* (et à travers lui beaucoup de mots en *.tion) est, comme on le sait, la clef de voûte statistique de ce discours en mouvement (figure 6.4), et nous remarquons que le réseau repère ici l'enchainement *pronom démonstratif + transformation* qui permet à Macron de surligner (la monstration, sinon la démonstration) d'innovations souhaitées [1]. Les *plateformes numériques*, renforcées par l'adjectif *grandes*, sont également typiques d'un président *high tech* qui promet le progrès et la modernité. À propos de renforcement, le modèle remarque encore *profonde* qui qualifie *désorganisation* : de fait, l'adjectif *profond* est l'adjectif préféré de Macron surtout quand il qualifie le mouvement (transformation profonde, innovation profonde..., ou refondation en profondeur, etc.) (figure 6.4). Et tou-

1. Nous ne pouvons développer ici. Dans un discours dynamique mais non abouti, Macron célèbre « la transformation » (sans complément de nom) ou utilise de manière intransitive le verbe « transformer » (sans complément d'objet : je veux transformer.). Dans ce cadre discursif qui fait de la transformation une finalité sans fin, le président semble souvent utiliser de manière rhétorique ou circulaire le démonstratif (cette transformation) : il feint ainsi de renvoyer à une transformation qui, précisément, n'aura jamais été définie auparavant dans le discours autrement que par un absolu. (Pour un développement pointu, nt. la monovalence du verbe « transformer », voir Macron ou le mystère du verbe. Ses discours décryptés par la machine. (MAYAFFRE 2021)

jours, car le passage repéré par la machine est magistral, la *souveraineté* (souveraineté numérique s'entend) signe la fin de l'extrait et apparait à l'analyse comme une préoccupation majeure des discours de Macron ; là où les *acteurs européens* en fin de passage renvoient de manière caricaturale, en un syntagme unique, à la double posture fondamental du macronisme : le pragmatisme affiché ou l'action entrepreneuriale (vs. l'idéologie ou la politique) symbolisés ici par le mot-clef *acteurs* (vs. les *citoyens* ou le *peuple* ou les *ouvriers*...), et sa foi dans l'Europe, de sa campagne électorale de 2017 jusqu'à sa politique vaccinale de 2021, ici verbalisée par *européenne*.

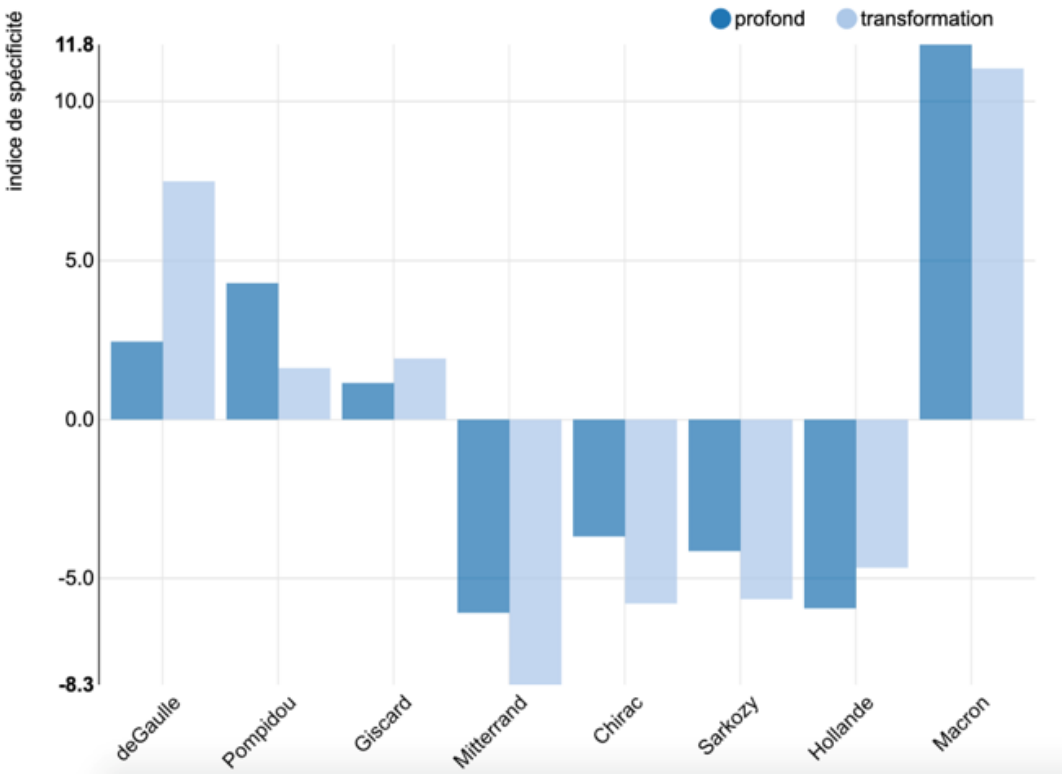


FIGURE 6.4 – Sur-utilisation statistique de *profond* et *transformation* par Macron

Les spécialistes d'Emmanuel Macron ainsi que ceux de logométrie remarqueront, au fond, que la plupart des mots ou codes grammaticaux activés par le réseau IA sont des mots statistiques de Macron, à l'image de l'illustration que nous avons donnée de *transformation* ou de *profond* (figure 6.4). Ce passage clé serait ainsi, peut-être, seulement une somme de spécificités. Pourtant, c'est bien la combinaison de ces mots (la convolution) c'est-à-dire leur contextualisation (on pourrait aussi dire leurs *cooccurrences* ou leurs *collocations* si l'on veut rester dans un cadre statistique) qui fait sens. Ces mots font texte chez Macron par leur engrenage sur la chaine, dans ce passage précis, et pas nécessairement ailleurs. Par exemple, le futur surligné par la machine n'est pas statistiquement très marqué, mais devient pertinent, chez Macron et dans ce passage, lorsqu'il s'agit de promouvoir la *transformation* ou de compenser une *désorganisation*. De même, *grandes* n'est aucunement un

adjectif macronien (il est plus utilisé par Giscard, par Mitterrand ou par Chirac), mais devient pertinent, chez Macron, pour qualifier la *révolution numérique* que le président de la *start up nation* appelle de ses vœux. C'est en cela que les mots et motifs remarqués semblent répondre à la définition de *passage* proposé par Rastier : ils font sens non pas en eux-mêmes pour eux-mêmes, mais à point nommé, dans le cadre d'un contexte d'utilisation – *parce que seuls les contextes sont constituants* (Rastier 2020) – , c'est-à-dire, plus généralement, dans le cadre d'un parcours de lecture, que la machine objective non seulement localement (les *unités* dans leurs environnements immédiats) mais globalement (puisque toute sortie-machine se trouve informée par le corpus d'apprentissage en général).

6.4. Généralisation des motifs profonds

Le *deep learning* sait s'adapter aux caractéristiques de chaque langue, et offrir un point de vue pertinent quelles qu'en soient les règles et les contraintes. Le latin et ses nombreuses variations qui lui sont spécifiques est un bon moyen de tester la généralisation de notre méthode. Voici un des extraits de Jules César les mieux reconnus par notre modèle de classification.

> *[...] ut tum accidit . MVNITIO quam pertinere a castris ad flumen SVPRA DEMONSTRO Adj :1Cl:Gen:Sing:Pos:MascNeutr Subs:3Decl:Gen:Sing cornus cohortes ignorantia loci sunt secutae cum portam quaererent cas- trorum que eam munitionem esse arbitrarentur . Subord Aux ANIMADVERTO coniunctam esse flumini prorutis munitionibus defendente nullo transcenderunt omnis que noster equitatus eas cohortes est [...]*

(Cesar, Belum ciuile, 3, 68)

La déconvolution *multi-channels* et le **wTDS** donnent ici les éléments de réponse qui poussent le modèle à reconnaître fortement l'auteur. Dans cet extrait, le *deep learning* reconnaît une occurrence d'un motif textuel qui a été clairement identifié par (D. Longrée et Mellet 2013) et (Lavigne et S. Longrée D. a. M. 2018). De simples variations du motif *as ut supra demonstraui* (littéralement : « comme je l'ai expliqué ci-dessus ») ou encore *quem antea memorauimus* (littéralement : « dont nous avons déjà parlé ») ont été déjà mises en évidence en utilisant nos outils d'ADT classiques. Cependant, jusqu'à présent ces méthodes n'ont jamais étaient capables de retrouver une variante aussi complexe que *munitionem, quam pertinere a castris ad flumen supra demonstrauimus* (le retranchement qui, comme nous l'avons expliqué plus haut, s'étendait du camp à la rivière), où le pronom relatif n'est pas le complément du verbe comme *in quem antea memorauimus*, mais le sujet d'une clause infinitive dépendent du verbe *demonstrauimus*. Le *deep learning* montre son efficacité pour identifier les saillances d'un texte ou un auteur en exploitant ici des structures multidimensionnelles (*multi-channels*).

Dans le même extrait, on peut aussi noter d'autres structures qui peuvent aussi être considérées comme un *motif* linguistique, *Quod cum esset animaduersum coniunctam esse flumini, prorutis munitionibus defendente nullo* (Et comme il avait été observé qu'il était relié au fleuve, les défenses ayant été jetées, personne ne le défendait). Ce motif

syntaxique est caractérisé par le mot *cum* suivi par deux ablatifs absolus (Mellet et D. Longrée 2009). De plus, ce *motif* inclut une occurence du *motif* multidimensionnel *quod ubi cognitum est* (lorsque cela a été connu) construit à partir du pronom relatif quod suivi d'une conjonction de subordination et d'un verbe à la forme passive (Lavigne et S. Longrée D. a. M. 2018).

Il semble avec ce dernier exemple, que la méthode permet donc d'entrevoir les *motifs* textuels définis notamment par (D. Longrée et Mellet 2013) comme « une construction lexico-grammaticale associée à un nombre restreint de formes, dont la fonction sémantique et discursive reste comparable d'une réalisation à l'autre ».

L'usage du deep learning descriptif tient donc ses promesses face à l'ADT classique. Les techniques d'apprentissages pronfonds revèllent des sutructures de texte complexes jusqu'ici difficiles à explorer avec la statistique seule. Les cas d'utilisation sont nombreux mais la prédiction (ou classification) de nouveaux textes (inconnus) reste un des points les plus fort de la méthode. Nous allons maintenant voir qu'en utilisant le **wTDS**, cette tâche que le deep learning maitrise par construction, peut être appliquée à des questions linguistiques sensibles telles que l'intertextualité.

Chapitre 7

Intertextualité

Ce chapitre reprend pour l'essentiel l'article Mayaffre *et* Vanni *2020. Des modificiations mineures et certains enrichissements ont été apportés pour structurer la présentation des différents résultats.*

7.1. Introduction

L'intertexte et l'intertextualité sont des concepts linguistiques majeurs mais fuyants depuis Kristeva, Barthes, Riffaterre ou Genette sinon depuis Bakhtine [1]. Ils apparaissent rétifs à toute forme d'objectivation, de formalisation, de mesure, c'est-à-dire, pour nous, à toute forme d'implémentation informatique définitive. Si tous les linguistes conviennent que l'intertextualité est une condition de l'interprétation d'un texte-cible, aucun ne prétend pouvoir clairement l'expliciter et la circonscrire au-delà de la convocation discrétionnaire de telle référence supposément éclairante ici, de tel discours prétendument inspirateur là, de telle reprise textuelle soupçonnée ailleurs. Sur les bases d'une définition aussi fragile [2], l'analyse statistique des données textuelles semble condamnée à l'impuissance méthodologique.

Nous considérons ici les corpus réflexifs numériques (Mayaffre 2002 ; Rastier 2004) [3] comme la matérialisation d'un certain intertexte du texte-cible et proposons un protocole méthodologique pour objectiver cet intertexte désormais matérialisé. Dans cette contribution, le texte-cible est le discours de Macron depuis son élection en 2017, et l'intertexte pressenti de ce texte-cible est l'ensemble du discours élyséen depuis 1958 (de Gaulle, Pompidou, Giscard, Mitterrand, Chirac, Sarkozy et Hollande) dans lequel le nouveau président peut puiser des références conscientes ou inconscientes, explicites ou implicites. On notera que cet intertexte pressenti – un intertexte parmi d'autres possibles – se justifie par des raisons génériques (discours officiel présidentiel), énonciatives (locuteurs dans une

1. Au-dela des auteurs majeurs et fondateurs évoqués ici, cette contribution s'appuie spécifiquement, du point de vue conceptuel, sur un numéro de revue (Cahiers de praxématique, 33, 1999) et un ouvrage collectif (Bres et al. 2005)

2. On se rappelle par exemple de la définition fondatrice mais évanescente de Barthes : « L'intertexte est un champ général de formules anonymes, dont l'origine est rarement repérable, de citations inconscientes ou automatiques, données sans guillemets » (*Encyclopaedia universalis, ed. 1995, Tome 22, p. 372*

3. Rappelons : un corpus réflexif est un corpus qui contient ses propres ressources interprétatives, c'est-à-dire dans lequel chaque texte constitue le contexte interprétatif de tous les autres, et l'ensemble des textes constitue le contexte interprétatif de chacun ; dans le but d'une interprétation « endogène » mieux contrôlée.

même situation d'énonciation), historiques (l'unité chronologique que constitue la Vème République) ou encore politiques (prétention centriste de Macron à être *en même temps* gaulliste et mitterrandien, sarkozyste et hollandais, giscardien et chiraquien).

7.2. Protocole

Sur un jeu de données de 1000 discours présidentiels sous la Vème République, avant la présidence Macron, équivalant 3 millions d'occurrences, le modèle décrit précédemment dans le chapitre 6, apprend à reconnaître (prédire) les discours de de Gaulle (phrases longues, plutôt nominales ou adjectivales, avec par exemple la « France » ou « l'état » comme premiers noms) comme ceux de Mitterrand (phrases courtes, plutôt verbales, avec le « je » et le « moi » comme centre d'intérêt personnel et « l'Europe » comme horizon). Il apprend à reconnaître les discours de Sarkozy (un lexique fort et une syntaxe simple) et ceux de Hollande (une syntaxe compliquée et un lexique affadi). Il apprend à reconnaître les textes de Giscard (phraséologie technocratique ou didactique), ceux de Chirac ou ceux de Pompidou (style littéraire, riche voire ampoulé) [4] ; et le corpus de validation (10% du corpus d'entrainement composé de discours présidentiels inconnus et anonymisés) permet de chiffrer que le modèle retrouve automatiquement au-dessus de 92,3% des fois le bon auteur-président des discours.

Ainsi, en apprenant à la machine le discours élyséen de de Gaulle à Hollande, entre 1958 et 2017, nous avons construit un certain « horizon d'attente » (selon la terminologie de la sémiologie des décennies précédentes), ou au contraire un certain point de départ des discours de Macron : selon nous, le modèle a appris un certain *intertexte* dans lequel Macron pourra emprunter pour construire ses propres discours ou duquel Macron pourra s'inspirer pour parler ; ou encore, un certain intertexte dans lequel l'auditeur puis l'analyste pourront faire résonner les discours de Macron pour les comprendre et les interpréter.

Puis, nous versons les discours de Macron, inconnus du système, dans ce corpus élyséen de référence (ou intertexte élyséen), en demandant à l'algorithme de rapprocher (classer) chaque segment de texte (ie. des fenêtres de 50 mots) de Macron d'un des présidents précédents qu'elle connait ; non sans artificialité donc, nous forçons le programme à attribuer (prediction) les paragraphes de Macron à un de ses prédécesseurs en raison de ressemblances linguistiques détectées. Ainsi, si Macron devait s'écrier à la tribune « Vive la Syrie libre ! », la machine attribuera ce passage à de Gaulle en référence sans doute au « Vive le Quebec libre ! » de Montréal en 1967. Si Macron devait prononcer « vous n'avez pas le monopole des sentiments », ou peut-être « vous n'avez pas l'exclusivité des sentiments », la machine attribuera le passage à Giscard d'Estaing en référence au débat télévisé avec Mitterrand en 1974 durant lequel le candidat conservateur avait répliqué au candidat socialiste « vous n'avez pas le monopole du coeur ». Pour donner un premier résultat réel du travail, lorsque Macron déclare «on ne peut pas travailler moins et gagner plus» [5], le modèle attribue automatiquement la phrase à Sarkozy en référence sans doute au « il faut travailler plus pour gagner plus » que le président de droite avait souvent répété durant son mandat.

4. Pour une analyse des discours et du style des présidents sous la Vème République voir Mayaffre 2012a ; Mayaffre 2021.

5. E. Macron, Voeux aux Français, 31 décembre 2018.

Enfin, dernière étape décisive pour l'étude linguistique, l'algorithme de déconvolution présenté dans le chapitre 3 et optimisé dans les chapitres suivants (4 et 6) permet de décrire le corpus : il s'agit non seulement d'extraire les phrases de Macron attribuées à de Gaulle ou Pompidou, Sarkozy ou Hollande, mais de surligner les éléments linguistiques qui ont participé, via les couches cachées du système, à la décision. Ici, la méthode d'intelligence artificielle qui sert de guide (convolution puis déconvolution, et indice de reconnaissance des unités saillantes du réseau : Text Deconvolution Saliency - **TDS** et **wTDS**) est doublée par les procédés logométriques traditionnels (l'historique calcul des spécificités (Lafon 1980) ou le calcul des cooccurrences), afin d'affermir les résultats de l'IA par la statistique textuelle.

7.3. Resultats

7.3.1. Prédiction

Les emprunts ou empreintes du discours macronien sont riches et variés ou, autrement dit, l'intertexte des discours de Macron est ample. Sur les 10.000 extrait (fenêtres de 50 mots) analysés du corpus Macron, les taux d'inspiration ou d'intertextualité – c'est-à-dire ces passages de Macron que l'algorithme attribue à de Gaulle, Mitterrand ou Hollande... – se hiérarchisent selon la figure 7.1.

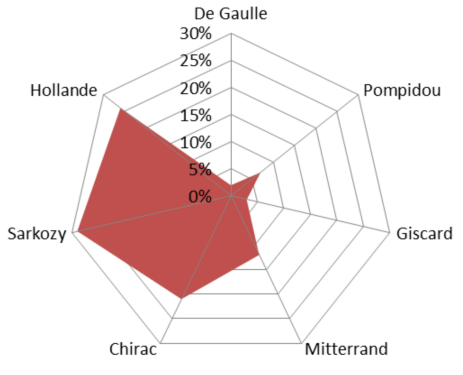


Figure 7.1 – Périmètre de l'intertexte de Macron

Conformément à la contrainte du temps qui pèse sur les séries textuelles chronologiques (Salem 1991, M. Guaresi, Mayaffre et Vanni, 2022 (sous presse)), Macron emprunte plus à ses prédécesseurs immédiats des années 1990-2010 (Hollande, Sarkozy, Chirac) qu'il n'emprunte à ses prédécesseurs plus lointains des années 1950-1970 (de Gaulle, Pompidou ou Giscard) ; la mémoire discursive est une mémoire de plus ou moins courte durée.

Dès lors, sur la base de ce constat chronologique, les hiatus attirent l'attention de l'analyste (Mayaffre 2021). Ainsi, à rebours de la chronologie, les empreintes pompidoliennes dans le discours de Macron sont plus nombreuses (7%) que les empreintes giscardiennes (3%) : par sa tenue littéraire (richesse et variété du vocabulaire), Macron imite souvent

l'Anthologiste de la poésie française ; un souffle pompidolien traverse souvent les discours du jeune président.

De même, en dépit de l'impératif chronologique, l'intertexte sarkozyste (29%) du discours de Macron est plus important que l'intertexte hollandais (26%) : ici ce chiffrage peut éclairer le débat sur l'identité discursive d'Emmanuel Macron et son positionnement politique fondamental peut-être plus à droite de l'échiquier qu'à gauche (Mayaffre, Guaresi et Vanni 2020).

Cependant, si la classification est intéressante – il s'agit d'une force reconnue du deep learning ici exploitée de manière originale pour mesurer l'intertexte – c'est la description linguistique du corpus qui constitue la plus-value méthodologique que nous cherchons à souligner ici.

7.3.2. Description

La force de l'algorithme présenté dans les chapitres précédents est de faire remonter, par déconvolution, les zones d'activation du réseau à un triple niveau que nous appelons *motifs profonds* et qui utilise les formes graphiques, les lemmes et les étiquettes morpho-syntaxiques associées aux mots. Rappelons surtout que les passages sélectionnés et les zones d'activation dans le passage doivent être considérés en contexte, c'est-à-dire non pas dans une logique occurrentielle (tokenisation et distribution quantitative de chaque token ou de chaque étiquette dans le corpus) mais dans une logique co-occurrencielle ou convolutionnelle (combinaison ou co-présence des éléments surlignés dans le passage).

A titre illustratif, les sorties machines de cette contribution sont issues du discours des voeux aux Français pour l'année 2020 d'Emmanuel Macron.

Exemple d'intertexte sarkozyste dans le discours de Macron

Dans ce discours donc, comme dans le corpus en général, certains passages sont jugés inspirés par Sarkozy, qui est, selon la classification globale (supra figure 7.1), le premier inspirateur des discours de Macron :

> *[...] travailler mieux, de partager la richesse créer dans toutes les NOM, à aider notre agriculteur et notre pêcheur à vivre dignement de leur travail comme à tout le entrepreneur et salarié. C'VER :pres DET :ART NOM PRP une Nation forte et indépendante. Si PRO :PER voulons lutter efficacement [...]*

*(Intertexte sarkozyste dans le discours de Macron. (Passage attribué à Sarkozy par l'algorithme qui souligne (**wTDS**) les lemmes (en vert), les formes (en bleu) et les étiquettes (en orange) responsables de la prédiction.)*

De fait, tout dans ce passage de Macron peut renvoyer au discours de Sarkozy. Au niveau lexical, l'algorithme souligne le lemme *travailler* (verbe) et le lemme *travail* (nom). Effectivement, statistiquement, il s'agit de deux grandes spécificités lexicales sarkozystes (et aussi macronistes) comme l'atteste la distribution quantitative des termes dans le corpus.

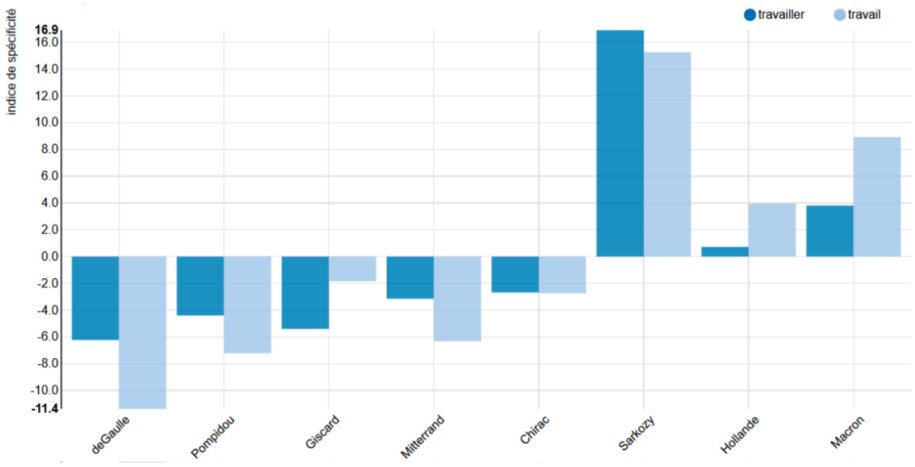


FIGURE 7.2 – Les lemmes *travailler* et *travail* dans le corpus présidentiel 1958-2020

Dans la logique convolutionnelle (i.e. contextuelle) dont on a parlé, on remarquera aussi dans le passage la combinaison de *travailler + mieux* : les adverbes mélioratifs sont ici aussi, à l'analyse, très fortement sarkozystes dans le corpus (en l'occurrence sous la forme *travailler plus*, *travailler davantage*, *travailler mieux*).

Mais à ce socle lexical autour du *travail*, s'ajoutent d'autres traits lexicaux sarkozystes qu'Hyperbase remarque, comme l'énumération de catégories socio-professionnelles que le président de droite aimait en son temps caresser : les *pêcheurs* ou les *agriculteurs* par exemple qui constituent électoralement la catégorie socio-professionnelle la plus fidèle à la droite avec les médecins, et encore les *entrepreneurs* et les *salariés*.

Ajoutons encore que dans ce passage, deux formes graphiques (en bleu) sur-lignées par le **wTDS** ne surprendront pas : *voulons* et *efficacement*. Macron emprunte en effet souvent à Sarkozy son volontarisme discursif, et son affirmation d'actions et d'efficacité. Du reste, c'est ce volontarisme verbal et cette rhétorique de la conviction qui expliquent sans doute le repérage du motif grammatico-lexical *C'+verbPrésent+Det+Nom*. Le pronom démonstratif neutre est la grande signature du sarkozysme en discours (MAYAFFRE 2012b). Ici, suivi d'un verbe conjugué au présent et suivi d'un nom (*c'est la France...*, *c'est un pays...*, *c'est le travail...*, *c'est la condition...*, etc.), le pronom bien nommé démonstratif donne une force de conviction qui n'est pas sans rappeler Sarkozy dans la bouche d'Emmanuel Macron.

Exemple d'intertexte hollandais dans le discours de Macron

Dans son discours de voeux aux Français pour l'année 2020, Macron reprend, à côté du discours de Sarkozy, le discours de Hollande, afin d'équilibrer politiquement son propos :

> *[...] d'un projet de justice et de progrès social. Un NOM de justice et de progrès social parce qu'PRO :PER assure l'universalité : il s'agit de faire en sorte KON*

un euro de cotisation versé ouvre les même droits pour tous dès la première heure de travail [...]

(Intertexte hollandais dans le discours de Macron. Passage attribué à Hollande par le modèle qui souligne (**wTDS**) les lemmes (en vert), les formes (en bleu) et les étiquettes (en orange) responsables de la prédiction)

Sans surprise, le passage repéré qui vise à justifier une réforme des retraites contestée par les syndicats, cultive un lexique de gauche : *progrès social*, *justice*, *universalité*, *les mêmes droits pour tous*. Par ces mots, l'écho intertextuel à Hollande et, à travers lui, au parti du mouvement sonne éloquemment. Par exemple, dans le corpus élyséen (1958-2020), *universalité* est bien une spécificité statistique de Hollande.

Et de droite et de gauche, le discours de Macron emprunte ainsi des deux côtés de l'échiquier. Dans un contexte politique difficile qui vise à satisfaire et l'électorat de droite et l'électorat socialiste (ou CFDétiste), Macron multiplie et diversifie ainsi les reprises phraséologiques, les emprunts lexicaux ou les clins d'oeil discursifs (Mayaffre 2021).

Exemple d'intertexte pompidolien dans le discours de Macron

Pour finir un extrait qui emprunte à un autre président plus éloigné dans le temps, mais remarquable selon l'I.A. si on compare le taux d'attribution de Macron à ses prédécesseurs du début de la 5ème république (figure 7.1).

[...] d'une NOM européenne. Restons ce peuple uni, ADJ, fier de son histoire, de ses NOM, de sa culture, confiant dans l'avenir et le progrès, sûr de son talent et de son énergie et ambitieux pour lui-même [...]

(Extrait de Macron attribué à Pompidou dans le discours des vœux du 31 décembre 2020)

À l'analyse, nous pouvons facilement argumenter que la prose macronnienne imite subtilement ici la prose pompidolienne. D'abord la phrase est très nominale comme le discours de Pompidou était le plus souvent nominal. Ensuite, Macron convoque la *culture*, le *progrès*, *l'histoire*, le *peuple l'avenir* : autant de mots que l'anthologiste de la poésie française, attaché à Malraux comme à de Gaulle, aimait convoquer dans des discours souvent épiques. La structure *NOM + européenne* (chez Macron sous la forme de *renaissance européenne*) est également une structure forte du discours de Pompidou qui parle plus que les autres dans le corpus élyséen de la *foi européenne*, *politique européenne*, *construction européenne*, *agriculture européenne*, *union européenne*, *identité européenne*, *défense européenne*. L'énumération d'adjectifs, encore, avec la mise entre virgules (*peuple uni, ADJ, fier*) se trouve être typiquement pompidolienne : ce type d'énumération participe à la richesse du discours qui caractérise dans le corpus élyséen le discours de l'agrégé de Lettres ; et que Macron cherche parfois à atteindre.

Dans les deux derniers exemples donnés, on remarque que Pompidou n'utilise pas le nom *renaissance* ni le syntagme *renaissance européenne*, et il n'utilise pas l'adjectif *soli-*

daire dans ses énumérations adjectivables, mais le modèle multidimentionnel (mot, lemme, code) fait abstraction de ces non-réalisations lexicales pour souligner pertinemment le motif grammatical *NOM+européenne* ou *peuple uni, ADJ, (fier).*

Ces échos intertextuels au sens élargi – c'est-à-dire au sens général de variations ou modulations d'un même motif ou d'une même structure indentifiables, et non pas au sens étroit de citations explicites ou de plagiats – ne surprendront pas : le genre ritualisé des *discours de vœux* et la charge institutionnelle et historique qui pèse sur les présidents, font que la prose de Macron, le 31 décembre, tend à s'inscrire dans des *déjà-dits* que la machine repère, et qui sont attribués – en fonction du corpus de référence analysé – à leur étymon textuel.

7.3.3. Au-delà des textes

Les humanités numériques ne consistent pas seulement dans la numérisation des humanités mais dans l'humanisation du numérique. L'historien, le géographe, le linguiste du texte ne saurait se contenter aujourd'hui de scanner ses archives, ses cartes ou ses corpus ; il aspire à revoir fondamentalement, à la lumière du numérique ses objets, ses protocoles et ses conclusions.

Cependant, humaniser le numérique ne signifie nullement anthropomorphiser la machine en lui prêtant on ne sait quelle sensibilité ou intelligence, ni quelques forces probatoires. Humaniser le numérique, c'est le dé-positiver et adopter face à lui, plus que jamais, une posture herméneutique.

Les sorties-machines, au fond, comme le sens, comme le texte, ne sont pas positives, mais sont elles-mêmes interprétation des données, et restent pour l'essentiel à interpréter. La machine ne conclut rien mais re-présente à l'esprit humain les objets éternels des humanités ; ces représentations numériques entrent avec dynamisme dans le cercle de nos pratiques interprétatives multi-séculaires.

En la matière, si l'IA peut constituer une révolution, c'est précisément parce qu'elle renonce à réifier, naturaliser, positiver l'objet pour nous interroger sur le processus, c'est-à-dire l'objectivation ; elle nous donne moins à voir le texte pour lui-même, qu'elle participe à la textualisation, via le numérique ; c'est-à-dire qu'elle constitue moins pour nous un instrument extérieur à l'objet déjà-là qu'une instrumentation de nos parcours de lecture qui échafaudent le sens. Le deep learning en effet apprend des données textuelles jusqu'à corriger (*backpropagation*) le point de vue initial – or c'est le point de vue qui crée l'objet. Reste alors à l'analyste l'essentiel et le dernier mot : apprendre de cet apprentissage numérique, et comprendre – prendre par devers soi – les nouveaux observables que la machine performe pour faire texte.

Chapitre 8

Applications

*Ce dernier chapitre présente les développements informatiques et les choix technologiques mis en oeuvre pour élaborer un deep learning descriptif utile à la communauté linguistique. Notament Hyperdeep (*Vanni, Corneli, *D.* Longrée *et al. 2020a) et DeepFLE (*Ruggia *et* Vanni *2021) sont deux des applications créées pour lesquelles nous reprennons en partie le contenu des articles cités.*

8.1. Hyperbase

Au-delà des textes, le développement du deep learning pousse certains logiciels à se réinventer techniquement pour permettre la cohabitation entre les méthodes statistiques classiques avec celles issues des réseaux de neurones profonds. C'est le cas d'*Hyperbase*, le logiciel historique d'ADT du laboratoire UMR7320 : Bases, Corpus, Langage, dont nous proposons ici une version revisitée qui intègre les dernières contributions en matière de deep learning descriptif.

8.1.1. Rappels historiques

Hyperbase a été créé à l'origine par un normalien, agrégé de lettres classiques, Etienne Brunet. Ce linguiste de formation est aussi un informaticien pionnier dans l'utilisation de la machine appliquée à l'analyse de données textuelles. En effet en 1967, il tape à la porte de la société IBM pour numériser ses premiers corpus littéraires sur cartes perforées et teste les premiers algorithmes d'analyses de données (figure 8.1).

Vingt ans plus tard, et fort de nombreuses expériences sur différents langages de programmation et supports, E. Brunet se tourne vers les microordinateurs pour créer en 1989 la première version du logiciel Hyperbase (figure 8.2). On doit alors le nom du logiciel en partie à la technologie qu'il utilise à l'époque : HyperCard [1]. Cet environnement de programmation développé par Apple rassemble les éléments d'une base de données avec une interface graphique similaire à la navigation hypertexte [2]. Cet environnement propice à la représentation métaphorique d'un livre numérique permet à E. Brunet de se familiariser

1. Commercialisé par Apple avec le Système 5 à partir de 1987

2. L'hypertexte est un concept de lecture non linéaire du texte qui permet de naviguer d'un passage ou d'un document à un autre. L'évènement déclenché au moment de la sélection par l'utilisateur d'un mot ou d'un bouton est aussi l'occasion de lancer quelques lignes de codes ou d'exécuter des algorithmes plus complexes. Ce procédé est la base de la programmation du logiciel historique par E. Brunet

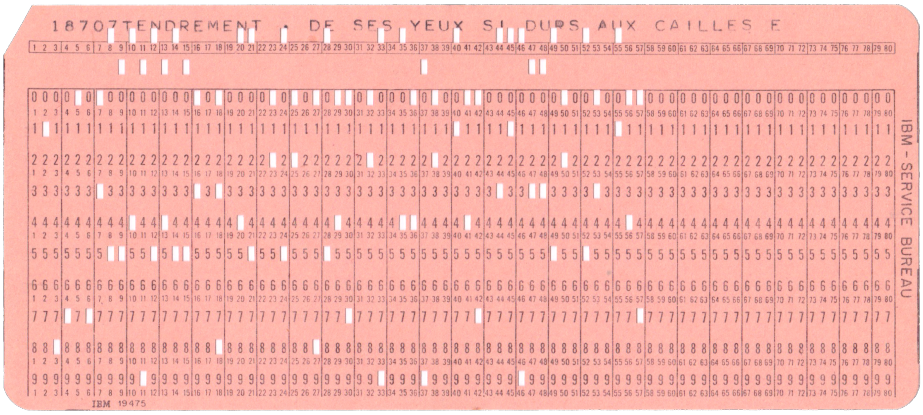


FIGURE 8.1 – « Il la considérait **tendrement, de ses yeux si durs aux cailles** et aux gelinottes » Extrait de *Juliette au pays des hommes* de Jean Giraudoux, numérisé par E. Brunet entre 1967 et 1978 (BRUNET 1978)

le concept d'interface homme-machine et de proposer finalement un outil complet mêlant analyses statiques et retour au texte dynamique. Associée à une documentation technique complète appuyée par de nombreuses publications, le logiciel commence à s'échanger sur disquette entre laboratoires. Les méthodes d'analyses de données textuelles numériques se démocratisent tout doucement.

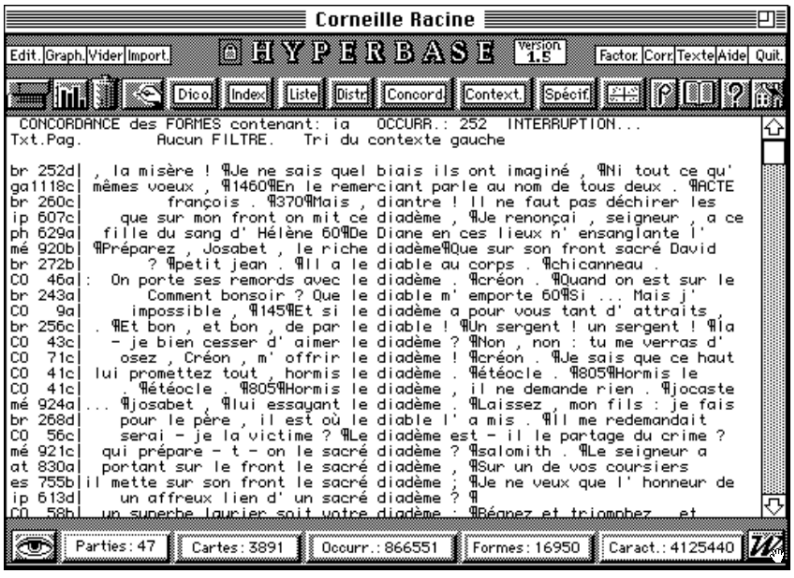


FIGURE 8.2 – Première version du logiciel Hyperbase en 1989

Avec l'abandon d'Hypercard par Apple vers la fin des années 1990, Étienne Brunet se tourne vers une autre technologie qui reprend la même idée de livre numérique : Tool-

Book. Produit est diffusé par Microsoft Windows, cet environnement permet de fabriquer des applications graphiques en utilisant le concept de page et de contenu. De plus le langage de programmation utilisé, OpenScript, est très similaire à HyperTalk le langage d'HyperCard. Cette technologie est la dernière en date utilisée par E. Brunet qui maintient son logiciel au fil des mises à jour de Windows et propose régulièrement de nouvelles fonctionnalités et du contenu (figure 8.3).

FIGURE 8.3 – Hyperbase (version 10)

La communauté autour du logiciel n'a cessé de grandir depuis ses débuts. Polyglotte et compatible avec les langues anciennes telles que le grec ou le latin, Hyperbase est aujourd'hui largement diffusé à travers le monde. Payant pendant un temps, il est devenu gratuit au fil des années et se télécharge depuis le site du laboratoire BCL [3]. Sensible à la popularité du logiciel et aux travaux portés par l'équipe qui l'utilise au quotidien [4] BCL a décidé depuis 2013 d'organiser la refonte d'Hyperbase et garantir ainsi un suivi des travaux et de la communauté. C'est ce travail de refonte et de développement inédits qui m'a été confié et dont cette section rend compte à grands traits.

8.1.2. Portage vers le web

Avec la démocratisation d'internet, les corpus et les outils ont évolué. Comme nous l'avons montré dans VANNI et MITTMANN 2016 le *traitement* a laissé place à la *visualisation* des données. Les big data ont poussé finalement certains logiciels à se réinventer. C'est le cas d'Hyperbase, confronté à d'autres logiciels utilisant des langages de programmation plus moderne et offrant des alternatives tout aussi performantes avec des visualisations graphiques et des parcours interprétatifs nouveaux.

3. `http://ancilla.unice.fr/hyperbase/`
4. Le laboratoire est aujourd'hui découpé en quatre équipes dont l'équipe logométrie qui participe à la diffusion du logiciel

Challenges

Le choix de la technologie s'est fait dans un contexte très prolifique : de nombreux logiciels gratuits téléchargeables et des interfaces graphiques toujours plus ergonomiques. D'autre part, le partage des données sur internet s'est imposé pour permettre l'exploitation des corpus à large échelle. Le lien entre le logiciel et la diffusion des données qu'il exploite s'est resserré.

Ces observations nous ont conduit à décliner Hyperbase en plateforme web. Aujourd'hui les technologies issues d'internet ne se limitent plus simplement à l'affichage de textes ou d'images, l'ergonomie des sites web respecte des standards qui évoluent naturellement au fil du temps et la notion de plateforme permise par des serveurs performants introduit des usages jusqu'ici réservés aux logiciels lourds [5]. Ainsi un outil tel qu'Hyperbase Web peut prétendre indexer de larges corpus, proposer des calculs statistiques avancés, permettre un partage des données et même plus récemment ouvrir la boite noire du deep learning aux humanités numériques. C'est le pari d'Hyperbase Web depuis 2014 (Vanni, Luong et Mayaffre 2014) puis d'Hyperdeep depuis 2020 (Vanni, Corneli, D. Longrée et al. 2020a) (module d'intelligence artificielle d'Hyperbase Web).

Les défis que se propose de relever Hyperbase Web sont donc nombreux : i) La mise à jour d'un logiciel historique sans dénaturer ses origines et conserver sa communauté d'utilisateurs ii) Un passage à l'échelle permanent qui autorise le stockage et l'indexation de corpus toujours plus volumineux iii) la diffusion d'outils modernes tels que le deep learning porté par une interface fluide et ergonomique.

Architecture materielle

Sans entrer dans les détails techniques (hors de propos ici), une simple présentation de l'organisation de l'architecture du code source donne un aperçu des moyens mis en oeuvre pour proposer aux linguistes en bout de chaine une I.A. spécialisée sur leurs données.

Le stockage des données et plus généralement l'hébergement du code principal sont répartis sur deux *datacenters* géographiquement distants. Le premier correspond à l'*équipement d'excellence* ORTOLANG, « un service spécialisé pour la langue, complémentaire de l'offre générale proposée par Huma-Num ». Ce serveur situé à Nancy, administré par l'Institut de l'Information Scientifique et Technique (INIST), héberge la version dite de production d'Hyperbase, c'est-à-dire le contenu tel qu'il apparait lorsque les internautes se connectent à l'adresse du site. Cette version est *stable* en termes de code et de qualité de service [6].

Le second centre est situé à l'université Nice Côte-d'azur, plus particulièrement dans les locaux du laboratoire BCL, il s'agit d'un serveur dimensionné pour le calcul (doté notamment de processeurs type GPUs) qui héberge la version de développement du code d'Hypebase. Ici des tests de nouvelles fonctionnalités sont faits et les bugs sont corrigés avant la mise en production. Bien que moins stable que la version de production, l'in-

5. ici la notion de logiciel lourd fait référence et s'oppose au client dit léger, sorte d'application entièrement gérée par un serveur web

6. la qualité de service (QDS) ou quality of service (QoS) désigne les conditions d'utilisation d'un service et plus particulièrement son accès réseau, sa disponibilité et la gestion du trafic des données

terface hébergée sur ce serveur reste accessible aux utilisateurs qui le souhaitent. C'est la version dite *beta* d'Hyperbase Web, qui héberge notamment le module de deep learing (Hyperdeep) toujours en développement. Les bases de données sont quant à elles recopiées régulièrement depuis le serveur de production vers le serveur de développement pour assurer la disponibilité et ajouter un niveau supplémentaire de redondance nécessaire à la tolérance aux pannes [7].

Ces différentes versions sont administrées via un gestionnaire de code source (GIT), qui autorise le partage de code et l'intégration de nouveaux développeurs tout en respectant les normes et les *workforkflow* classiques (au minimum un responsable - *intégration manager* - pour superviser les développeurs qui déposent leur code sur un serveur de dépôt privé).

L'administration système et réseau d'Hyperbase web est devenue avec le temps une tâche importante dans la chaine de production qui garantit des services pointus aux chercheurs. La plateforme atteind depuis 2021 plus d'1,6 milliard de mots répartis sur plusieurs milliers de bases de données multilingues créées par les utilisateurs. Hyperbase web est aujourd'hui utilisé dans le cadre de la recherche, mais aussi pour l'enseignement en France et à l'étranger.

Architecture logicielle

Cette plateforme s'appuie sur les standards en matière de développement logiciel qui distingue les différents éléments de l'architecture tels que la base de données, le moteur principal ou encore l'interface graphique [8].

La base de données repose sur un système de table de hachage type clés/valeurs pensées pour être distribuées sur plusieurs sites. La structure des données choisies est le JSON (*Javascript Object Notation*) pour sa portabilité vers les technologies du web. Aucune base relationnelle n'est utilisée (type SQL), Hyperbase se focalise sur les notions de corpus, textes et mots sans intégrer d'information sur les utilisateurs qui les administrent. Il n'y a donc pas de profil lié à une personne, pas d'inscription, simplement des bases de données publiques ou privées protégées par un nom unique et un mot de passe.

La partie métier du code (le contrôleur) utilise le langage python pour sa large communauté dans le domaine du traitement automatique des langues et les librairies associées. Cependant de nombreuses fonctionnalités spécifiques à Hyperbase et à l'analyse de données textuelles nécessitent une implémentation particulière. C'est le cas notamment l'analyse arborée (Vanni, Luong et Mayaffre 2014) ou du traitement de la cooccurrence (Vanni et Mittmann 2016).

7. La redondance des données est un élément clé de l'administration d'un système. En général chaque serveur dispose de son propre mécanisme de redondance (multiples disques durs - RAID). Mais l'hébergement sur plusieurs sites ajoute un niveau de sécurité supplémentaire, par exemple en cas d'incendie sur un des sites comme cela s'est produit avec le fournisseur OVH en mars 2021.

8. Ce type d'architecture utilisé pour le développement d'hyperbase web est aussi connu sous le nom de Modèle/Vue/Controlleur (MVC), même si en réalité cette architecture ce décompose en plusieurs autres sous-composants

Enfin l'interface graphique (la vue) utilise les standards actuels du web, avec le couple HTML/CSS + Javascript et différentes librairies qui répondent aux besoins en termes de visualisations graphiques (comme d3js par exemple) ou d'ergonomie. La communication entre la vue et le contrôleur se fait en utilisant une interface de programmation d'application (API) de type REST (*representational state transfer*) qui renseigne sur l'état de la requête et transfère de données (toujours en utilisant le format JSON). Cette méthode est particulièrement utile pour la programmation d'applications annexes (voir section 8.3).

8.1.3. Interface

L'interface graphique du logiciel historique a été repensée pour correspondre aux critères du web. Ainsi le point d'entrée de chaque base (une fois chargée) est un système de requêtes inspiré des moteurs de recherches standards sur internet. L'interface semble minimaliste au premier abord, mais la plupart des outils statistiques et documentaires proposés par Hyperbase sont là (figure 8.4).

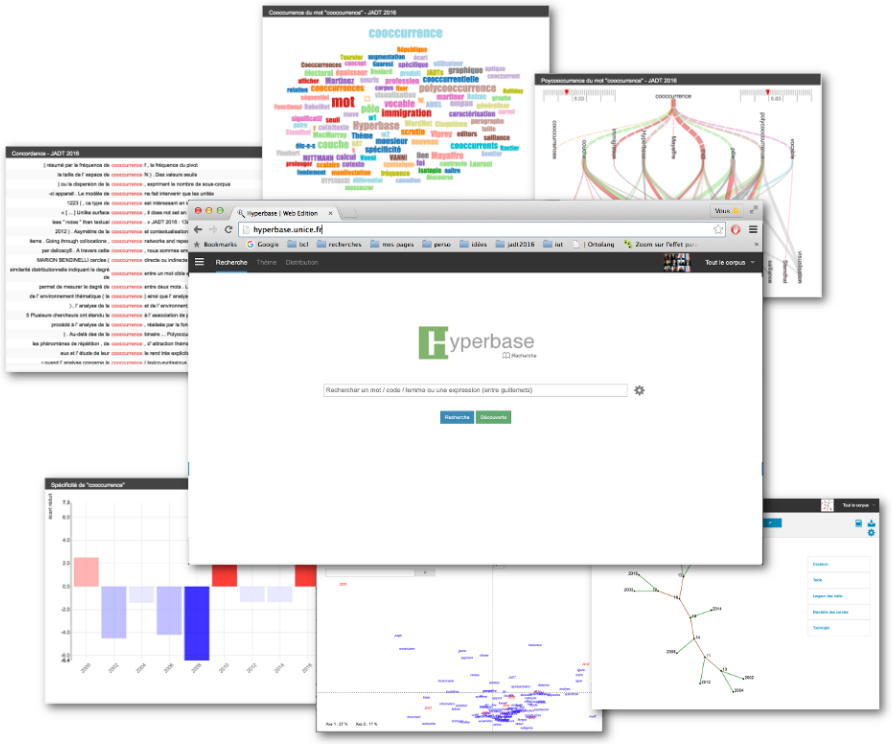


FIGURE 8.4 – Hyperbase Web : `http://hyperbase.unice.fr`

Les expressions régulières sont permises et la syntaxe favorise la recherche de motifs linguistiques composés en général de succession mots utilisant plusieurs niveaux de représentation : la forme graphique, le code grammatical et le lemme. La combinaison de ces trois niveaux ainsi que l'utilisation de quelques *jockers* permet de répondre à la majorité des recherches complexes.

Par défaut, l'interrogation du moteur donne lieu à un retour au texte immédiat sous forme de concordancier, mais il est aussi possible de manipuler la vue pour afficher différentes formes de contextes. Une fois la requête établie, le passage aux calculs statistiques se fait en navigant sur plusieurs onglets. Le premier *thème* reprend une fonctionnalité historique d'Hyperbase : le calcul de la cooccurrence (autour du mot ou de l'expression). Il est ici étendu jusqu'à la *polycoccurrence* associée à une visualisation graphique nouvelle (Vanni et Mittmann 2016). Le deuxième onglet renseigne sur la distribution des mots/expressions dans le corpus. Hyperbase Web comme son prédécesseur fonctionne sur des corpus contrastifs, c'est-à-dire des ensembles de textes regroupés par métadonnée et comparables entre eux. Cette distribution autorise des calculs de corrélations et met en évidence la surutilisation (indice de spécificité) d'un mot ou d'une expression dans une ou plusieurs parties du corpus. Le calcul de la distribution peut aussi être exploratoire lorsqu'il est utilisé sur les codes grammaticaux. Il est alors possible de demander la liste des mots les plus fréquents dans le corpus pour observer leur comportement au fil des textes. Des analyses plus complexes viennent alors accompagner l'interprétation du chercheur avec par exemple l'Analyse Factorielle des Correspondances (AFC) ou des médodes de *clustering* comme l'analyse arborée.

L'approche exploratoire est un aspect important d'Hyperbase. Comme pour la version originale du logiciel, il est possible d'explorer le corpus et de récupérer des indices statistiques qui renseignent sur le contenu des textes et le comportement des métadonnées. Ces fonctionnalités sont accessibles via le menu *lecture* et donne accès à une interface qui mêle le texte brut avec des indices statistiques tels que la distance intertextuelle (C. Labbé et D. Labbé 2003) ou le calcul des cooccurrences généralisées (*correlats*).

Le menu principal donne également accès à la gestion de la base (information concernant le contenu, partage des données mises à jour) et au partitionnement dynamique du corpus. Enfin ce menu ouvre sur Hyperdeep, le module deep learning qui implémente les travaux présentés dans les chapitres précédents et l'applique au corpus sélectionné.

8.2. Hyperdeep

Hyperdeep est un module à part dans le noyau d'Hyperbase. Il est distribué via son propre serveur de dépôt de code source et fait appel à un environnement particulier qui sollicite l'utilisation de processeurs types GPU.

8.2.1. L'entrainement deep learning

Ce module est accessible sur la version beta d'Hyperbase Web depuis le menu déroulant et propose, comme point d'entrée, de lancer un apprentissage sur le partitionnement courant. L'architecture des réseaux de neurones profonds repose, nous l'avons vu, sur un ensemble de paramètres qui modifient la complexité ou la profondeur des réseaux, les hyperparamètres. Ces paramètres peuvent potentiellement affecter la qualité de l'apprentissage. La plupart (bach size, learning rate, epoch, ...) sont accessibles via les options de l'interface graphique, mais sont en général réservés aux utilisateurs avertis du deep learning (figure 8.5).

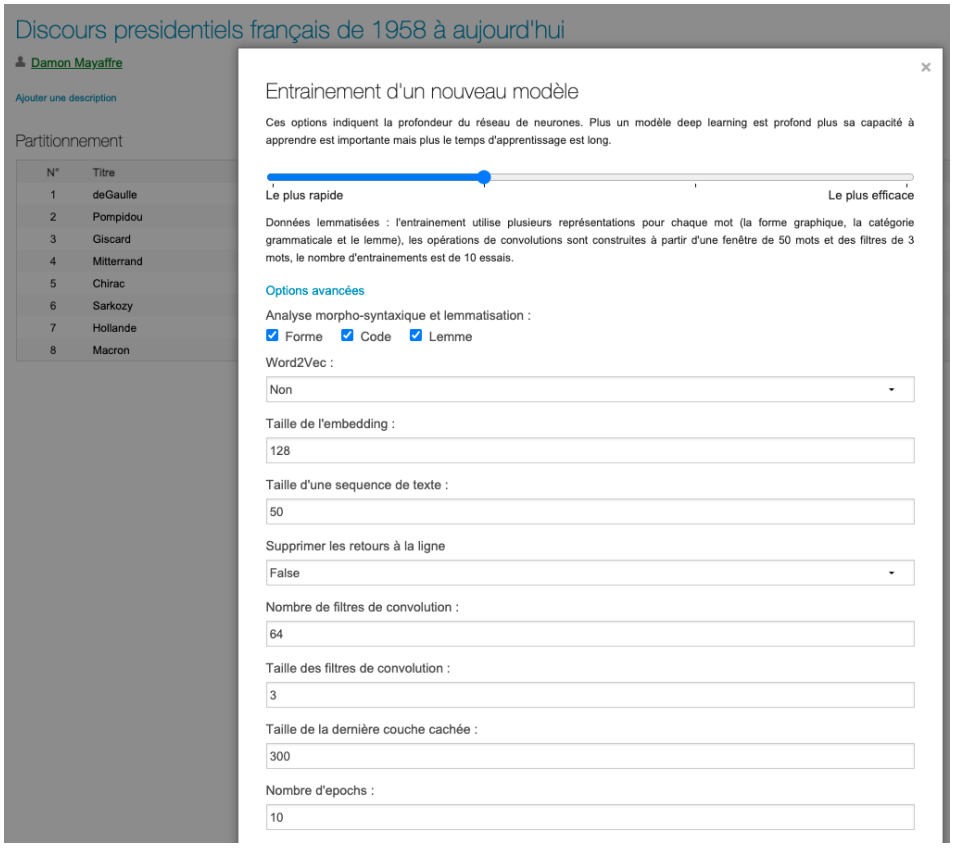


FIGURE 8.5 – Interface pour l'entrainement d'une base

Cependant que l'utilisateur soit averti ou non, il n'existe pas de règle ou de méthode pour prédire le paramétrage optimal de chaque modèle. L'interface permet de tester, corriger et retester encore jusqu'à atteindre un état stable convenable. Néanmoins, certaines bonnes pratiques existent et peuvent être appliquées pour modifier raisonnablement ces paramètres. Beaucoup d'articles existent à ce sujet (Aghaebrahimian et al 2019), nous ne détaillerons donc pas ici comment paramétrer un réseau deep, mais il faut noter que le taux de précision d'un modèle par défaut peut significativement évoluer en modifiant les options d'Hyperdeep avant de lancer un l'apprentissage. Une fois l'apprentissage terminé le score de précision du modèle est affiché (accuracy et loss) et l'ensemble des fonctionnalités d'Hyperdeep est activé.

8.2.2. Affichages des résultats

Prediction

La première fonctionnalité disponible après un entrainement est la classification de texte, une prédiction de l'appartenance d'un texte nouveau (inconnu du système lors de l'apprentissage) à l'une des classes. Hyperdeep fonctionne par défaut avec un simple copier-coller, mais il est possible de charger un fichier texte volumineux qui sera alors analysé dans son ensemble. Comme les réseaux convolutionnels fonctionnent sur des données de taille fixe

(donnée spatiale statique, voir section 2) le texte présenté sera découpé en segments de taille fixe, la longueur de ces segments étant un des nombreux hyperparamètres à choisir avant l'apprentissage. Chaque segment est ainsi attribué indépendamment des autres à une des classes. Et la somme de ces attributions donne un score global réparti entre toutes les classes. Cette répartition donne une idée du profil général du texte. Pour accompagner l'interprétation de cette classification, une visualisation graphique est proposée avec un diagramme de type radar qui permet d'avoir un aperçu rapide de l'orientation d'un texte (figure 8.6).

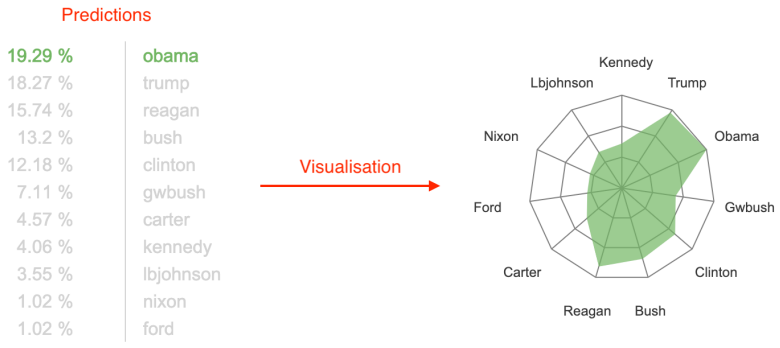


FIGURE 8.6 – Visualisation des prédictions dans Hyperdeep

Cette tâche de prédiction est naturelle pour la communauté du deep learning. Différents cas d'utilisation en SHS ont été présentés dans les chapitres précédents. Par exemple il est possible d'analyser un nouveau texte pour l'attribuer aux différentes classes (attribution d'auteur par exemple), ou encore de retirer une partie de corpus (un auteur) pour le soumettre à la prédiction et apprécier l'intertexte (chapitre 7). Et il y a certains cas qui ne sont possibles que par le biais du calcul des passages-clés et en utilisant la plus-value descriptive du deep learning (permis par **wTDS**). à savoir, redonner tout le corpus d'entrainement à la tâche de prédiction pour avoir une description fine des caractéristiques linguistiques fortes de chaque classe identifiées par les couches cachées du réseau. Nous allons voir maintenant comment exploiter cette description dans Hyperdeep.

Description

Le calcul du **wTDS** attribue un score à chaque mot (chaque token) pour chaque classe. Sa participation à la prise de décision finale peut être ainsi évaluée et comparée dans le contexte du partitionnement du corpus. Et avec l'utilisation de la lemmatisation et de l'étiquetage morpho-syntaxique proposée par Hyperbase et réutilisée par Hyperdeep, chaque mot correspond en fait à trois calculs de **wTDS**, un pour chaque représentation du mot : la forme graphique, le code grammatical et le lemme. Ces représentations sont entièrement contextualisées par la convolution qui applique une fenêtre glissante sur le texte et qui les regroupe dans les dernières couches du réseau. Le **wTDS** du mot est donc dépendant de son contexte et un passage devient clé pour le réseau au regard de l'ensemble des informations (forme/code/lemme) présent dans le passage.

Pour chaque token, le **wTDS** peut être soit positif soit négatif selon la classe observée en sortie et selon si le token a servi ou au contraire desservi cette classe. Pour capturer l'ensemble de ces informations et proposer un parcours interprétatif intuitif au chercheur, nous proposons une visualisation dynamique du texte (voir figure 8.7). Trois couleurs ont été choisies, bleu, orange, vert pour distinguer respectivement la forme, le code et le lemme. Par défaut un seuil de significativité est choisi pour n'afficher que les **wTDS** les plus élevés (à la manière du seuil de spécificité de la loi normale par exemple), mais ce seuil peut être ajusté par l'utilisateur au moyen d'un curseur qu'il peut déplacer pour chaque représentation du mot. Le côté réellement dynamique de cet affichage vient du fait que l'on peut survoler chacune des classes pour voir s'afficher les **wTDS** les plus élevés de la classe sélectionnée. Même si un passage est attribué à une classe, chaque mot a un poids différent selon la classe observée (voir chaptitre 4). Cette méthode permet de repérer notamment les caractéristiques partagées (ou pas) par plusieurs classes en même temps et de comprendre finalement pourquoi le passage a été attribué ou pas à une classe.

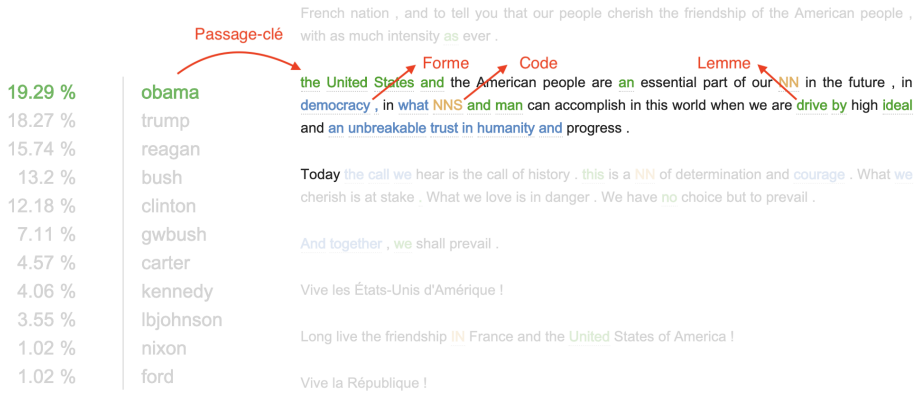


FIGURE 8.7 – Description et passages-clés dans Hyperdeep.

Nous allons maintenant voir deux cas concrets d'utilisation d'Hyperdeep sur deux langues différentes, l'anglais et le latin qui présentent toutes deux certaines caractéristiques différentes que le réseau deep learning est capable d'apprécier.

8.2.3. Exemples d'utilisation

Trump, Obama... Macron : les résonances du discours

En s'adressant, en anglais, au Congrès des Etats-Unis le 25 avril 2018, Emmanuel Macron réalise un exercice difficile. Le protocole diplomatique, les contraintes politiques, la bien-saillance exigée de l'invité contraignent évidemment le discours. Et la presse américaine est suspendue à la tonalité d'un discours qui sera jugé favorable ou défavorable aux Démocrates ou aux Républicains.

Les couches cachées du deep learning et le traitement statistique permettent de montrer comment le discours du président français s'applique à reprendre majoritairement le discours démocrate d'Obama pour faire passer un message universaliste et écologique, tout

en reprenant la forme du discours actuel des Républicains et de Trump.

Dans un vaste corpus d'apprentissage composé des discours des présidents américains depuis Kennedy, l'algorithme indique ainsi que le discours de Macron emprunte d'abord à Obama (19.29%). Par exemple, un des passages clés attribué à Obama résonne de mots que le président démocrate aimait en effet utiliser et que le **wTDS** surligne (figure 8.8). La thématique écologique renvoie explicitement à la présidence Obama (et s'oppose au refus de Trump de ratifier les accords de Paris). Statistiquement, par exemple, les mots *climate*, *planet* ou *transition* sont, de fait, caractéristiques d'Obama dans l'histoire des discours présidentiels américains (figure 8.9).

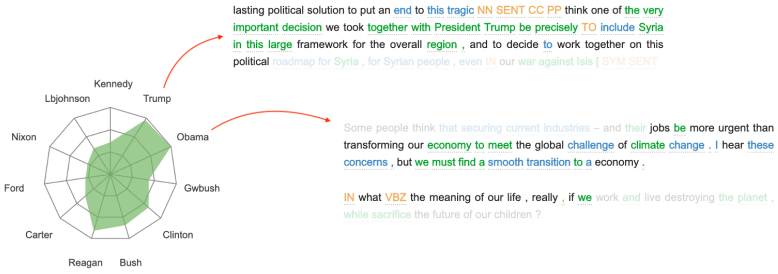


FIGURE 8.8 – Passages-clés attribués à Trump et Obama dans le discours de Macron.

Le **wTDS** peut être confirmé par la distribution statistique des observables du passage : par exemple, comme son hôte américain aime le faire, Macron utilise ici des formules superlatives comme « very important », et construit son discours autour de relances syntaxiques de type « NN (nom) SENT (ponctuation forte) CC (conjonction de coordination) PP (pronom personnel) » (*...work. And I...*, *...Nation. And you...*, *...country. And we...* etc.) (figures 8.8 et 8.9)

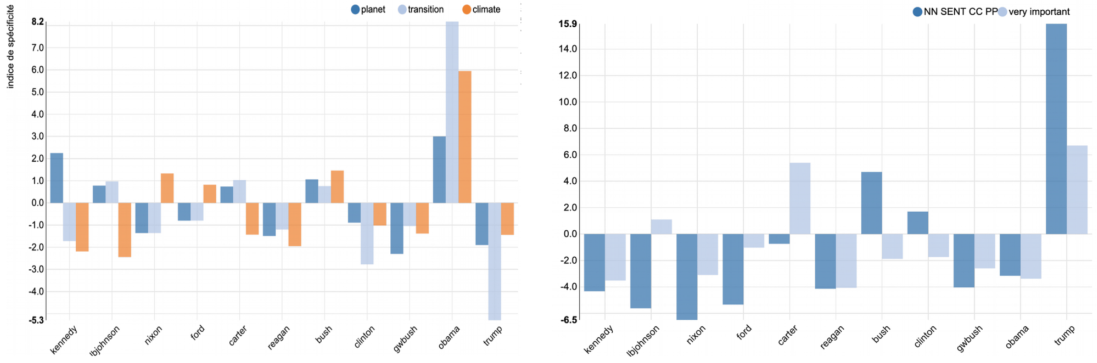


FIGURE 8.9 – Distribution statistique des marqueurs identifiés par le **TDS** dans le corpus anglais

Par hypothèse nous supposons que Macron, comme tout locuteur, construit son discours en empruntant consciemment ou inconsciemment à d'autres locuteurs (ici aux présidents

du pays hôte). Le logiciel présenté repère ces emprunts ou ces empreintes. Il le fait tant d'un point de vue lexical (particulièrement lorsque les discours sont proches chronologiquement et que, en conséquence, les thématiques sont avoisinantes) que d'un point de vue grammatical (particulièrement lorsque les discours sont éloignés chronologiquement, que le lexique conjoncturel s'est évanoui, et que seules les permanences grammaticales demeurent).

Les Héroïdes du poète latin Ovide, entre élégie et épopée

Le poète latin Ovide, contemporain de l'empereur Auguste, a produit une œuvre littéraire abondante et variée, tantôt relevant de l'élégie amoureuse, tantôt portant sur des sujets à caractères mythologiques ou historiques. On ne s'étonnera donc pas de voir qu'Hyperdeep, entraîné sur un corpus de poètes latins classiques, n'attribue pas massivement l'œuvre d'Ovide à l'un de ceux-ci, mais la distribue assez largement entre ceux-ci. L'empreinte de deux auteurs se fait toutefois sentir plus massivement, celle de Properce en premier lieu (32,41%), ensuite celle de Virgile (23,44%) (Figure 8.10). La chose s'explique aisément quand on sait que la production du premier consiste essentiellement en des poésies amoureuses et celle du second en une vaste épopée mythologie centrée autour de l'histoire du héros troyen Enée. Des passages d'une seule et même œuvre sont toutefois perçus par Hyperdeep comme étant particulièrement représentatifs de Properce, d'autres de celle de Virgile. Il s'agit en l'occurrence de passages de lettres fictives appelées *Héroïdes* que diverses héroïnes de la mythologie gréco-romaine adressent à leurs amoureux qui les ont délaissées. L'oeuvre comporte donc bien à la fois une composante amoureuse et une dimension mythologique. Ce que l'on notera toutefois avec intérêt, c'est que l'attribution d'un passage soit à Properce, soit à Virgile repose sur des facteurs sensiblement différents. C'est sans étonnement que l'on notera que les passages attribués à Virgile le sont essentiellement sur base des lemmes (en vert).

Dans l'exemple de Virgile (Figure 8.10), on rencontre un grand nombre de lemmes spécifiques à l'oeuvre (figure 8.11) : la coloration virgilienne du texte d'Ovide repose ici essentiellement sur le contenu du texte et sur la thématique mythologique et épique (marquée par les personnages, Enée, Mézence, les Rutules ou par une action telle que *arma induere*, revêtir ses armes) . Toutefois, les séquences de lemmes jouent ici un rôle déterminant : *MEZENTIVS INDVO ARMA* se rencontre une fois chez Ovide , alors que chez Virgile ont trouve *INDVO ARMA MEZENTIVS*.

Dans le cas de l'attribution à Properce de passages des Héroïdes, les codes grammaticaux semblent avoir un rôle plus important, comme le montre le deuxième exemple de la figure 8.10, où la reconnaissance ne s'appuie pas seulement sur des lemmes (*EGO* : je) ou des formes (*puella* : jeune fille), liés à un échange amoureux, mais aussi sur 3 codes.

La spécificité de ces codes a très certainement ici joué une rôle, puisque ils ne sont spécifiques que de Properce, avec des excédents significatifs, alors que les trois sont en déficit chez Virgile (figure 8.11).

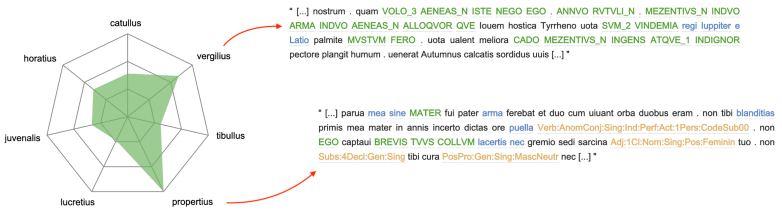


FIGURE 8.10 – Passages-clés d'Ovide attribués à Virgile et Properce.

Mais ici aussi la séquentialité des éléments a dû intervenir : en effet, les deux derniers des trois codes en orange dans le passage ci-dessus n'apparaissent séparés par 5 mots que chez Ovide et Properce. Le rôle joué tant par les codes que par leurs séquences s'explique ici fort bien : la parenté entre le deux textes ne repose pas seulement sur le contenu lexical des œuvres mais aussi sur diverses caractéristiques grammaticales propres à leur forme, comme l'emploi de formes de 1ère personne. Dans l'exemple d'Ovide, l'utilisation d'Hyperdeep permet d'objectiver ce que le philologue ne pouvait que pressentir à la lecture du texte.

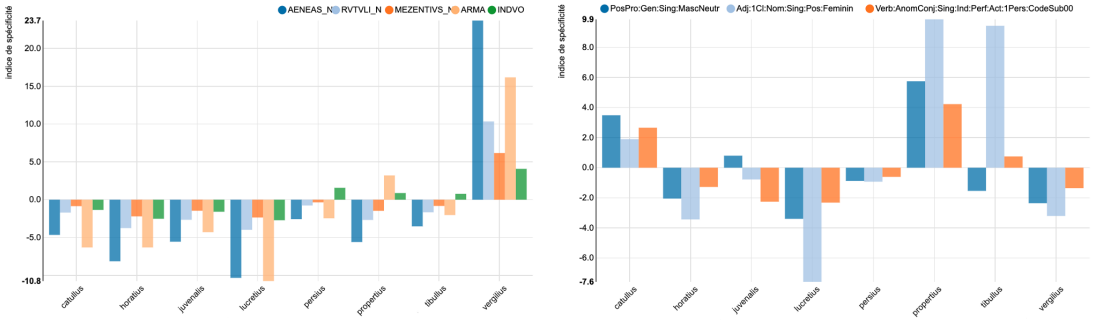


FIGURE 8.11 – Distribution statistique des marqueurs identifiés par le **TDS** dans le corpus latin

Autres fonctionnalités

Hyperdeep intègre d'autres fonctionnalités qui utilisent l'Embedding des mots avant (préapprentissage type word2vec) ou après l'apprentissage une fois que le modèle a corrigé cette représentation (voir section 2.5). L'embedding est un bon moyen de s'intéresser à la charge sémantique des mots dans le réseau de neurones. En effet il faut garder à l'esprit que lorsque la convolution applique une fenêtre glissante sur trois mots c'est en fait une convolution sur trois vecteurs numériques qui positionne chaque mot dans un espace à n dimensions. Au départ le préapprentissage contraint cette représentation pour que les distances euclidiennes reflètent la proximité sémantique des mots. Mais comme nous l'avons vu dans la section 2.5, après l'apprentissage du modèle, ce n'est pas toujours le cas. Un aperçu du nouvel espace généré est alors fort utile. Hyperdeep propose donc d'afficher les plus proches voisins d'un mot (code ou lemme) donné. Cette fonctionnalité que nous

appelons *thème* en référence à la fonction statistique du même nom est un outil précieux lorsque l'interprétation du **wTDS** seul n'est pas suffisante.

Cette fonctionnalité basée sur l'embedding ouvre sur une autre fonction, *correlats*, la version deep learning du *correlats* connus dans Hyperbase qui donne une représentation graphique des relations entre les mots les plus fréquents du corpus. Ce dernier outil met en évidence les variations qui peuvent exister d'un modèle à un autre qui utilise des hyperparamètres différents. C'est donc aussi une excellente aide à l'interprétation des résultats d'une classification.

Conclusion

Le deep learning semble avoir des qualités aussi bien prédictives que descriptives. Cette approche euristique qui suit la même philosophie que la plupart des outils d'ADT donne un grain à moudre d'un genre nouveau aux chercheurs. La boite à outils statistiques jusqu'à présent disponible se voit donc enrichie avec Hyperdeep d'une nouvelle méthode, un nouveau regard sur les textes complémentaire des approches existantes. Le deep learning n'est plus boite noire, il est maintenant verbeux, très verbeux, peut-être trop à tel point que la statistique et l'expertise linguistique deviennent les alliés nécessaires pour une prise en main objective de ces nouveaux outils. C'est dans cette idée qu'Hyperdeep prolonge l'expérience proposée depuis plusieurs décennies par Hyperbase à travers une interface nouvelle qui fait le pari de marier statistique et deep learning dans un même outil.

8.3. Autres applications

Pour clore ce chapitre sur les applications, notons que l'API REST permet aussi d'interroger Hyperbase Web et Hyperdeep depuis des applications tierces (un des intérêts majeurs d'une API). Les fonctionnalités d'Hyperdeep se retrouvent donc utilisées par d'autres applications qu'Hyperbase.

8.3.1. Mesure-du-discours

Le premier exemple est un observatoire du discours politique français, mesure-du-discours [9] (figure 8.12), véritable bac à sable pour l'application du deep learning en SHS qui a été introduite par les travaux de Ducoffe et al., 2016 et a fortement contribué à construire une preuve de l'intérêt de la méthode en SHS. Mesure-du-discours est en quelque sorte une des premières utilisations concrètes du *deep learning* et plus particulièrement du *political discourse learning* [10] (Mayaffre, Bouzereau et al., 2017)

Cette plateforme a été depuis étendue à l'ensemble des discours politiques traités par le laboratoire BCL, depuis la Révolution française jusqu'aux discours de l'actuel président E. Macron. En plus de l'analyse du profil discursif des différents acteurs politiques français, elle est aussi utilisée pour mettre en évidence les clivages politiques entre la gauche et la droite. Le deep learning peut identifier et décrire avec le **wTDS** les caractéristiques qui marquent les oppositions politiques comme le montre M. Guaresi 2020. Cette étude

9. `http://mesure-du-discours.unice.fr`
10. Projet IDEX UCAJEDI portant la référence ANR-15-IDEX-01

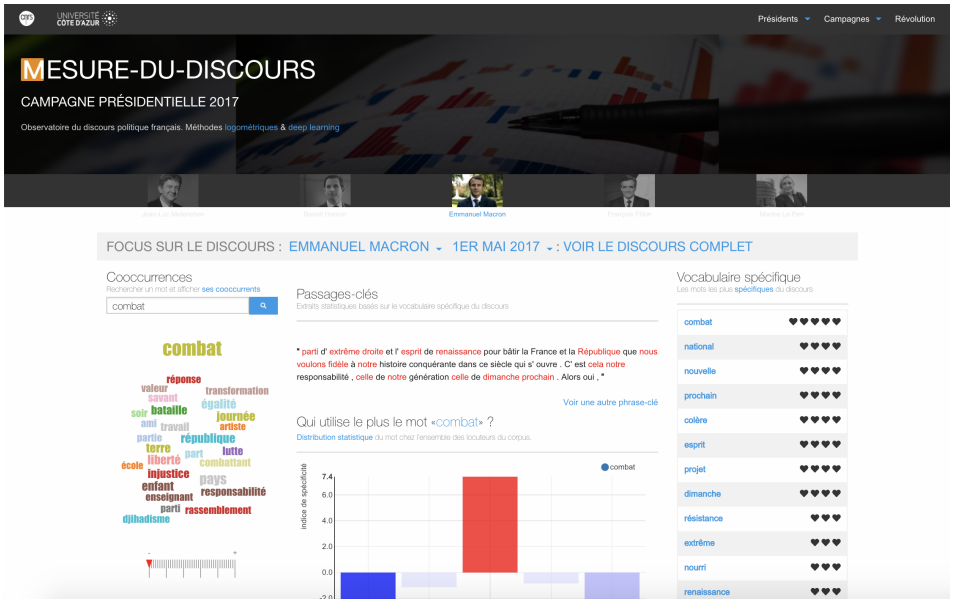


FIGURE 8.12 – Mesure-du-discours : `http://mesure-du-discours.unice.fr`

s'appuie sur large corpus de discours étiquettes à gauche et à droite sous la Vème république et montre la plus-value d'un deep learning descriptif pour la recherche d'observables linguistiques nouveaux en politique.

8.3.2. DeepFLE

Nous l'avons vue au fil des chapitres, les sujets traités par la méthode proposée s'étendent de la littérature (BRUNET et VANNI 2019) à la politique (MAYAFFRE et VANNI 2020) en passant par la linguistique formelle. Cette dernière est illustrée maintenant avec une autre application indépendante d'Hyperbase appliquée à la didactique du Français Langue Etrangère (FLE) (RUGGIA et VANNI 2021).

Les questions d'évaluation et de description des niveaux de langue selon les échelles du Cadre Européen Commun de Référence pour les langues (CERCL) (Conseil de l'Europe, 2001, 2018) touchent tous les acteurs du Français Langue Étrangère (FLE). Ainsi, dans la continuité des travaux de RUGGIA 2019, un outil a été créé capable à la fois de prédire le niveau d'un texte et de fournir une analyse descriptive de ses spécificités, DeepFLE [11] (RUGGIA et VANNI 2021).

Cadre méthodologique

Il s'agit d'une approche tout à fait singulière et novatrice dans le champ de la didactique des langues, actuellement du FLE, du russe et de l'anglais. L'hypothèse est ici que le "**wTDS** est capable d'extraire les caractéristiques de textes en français et plus précisément il est capable d'extraire les saillances qui marquent un changement de niveau selon

11. `http://deeptext.unice.fr/FLE`

le CERCL" (Ruggia 2019).

Techniquement pour pouvoir exploiter le **wTDS**, la platefrome utilise une base de données textuelles qui constitue le corpus d'apprentissage nécessaire pour l'apprentissage profond. Ce corpus est constitué de 6 classes de niveaux : A1, A2, B1, B2, C1, C2. qui totalisent 595.980 mots. Une fois entraîné avec Hyperdeep et comme pour les précédentes applications, la platefrome permet de lancer des analyses prédictives et descriptives en soumettant un texte au système. Cette dernière fonctionnalité permet de visualiser le pourcentage de reconnaissance ainsi que les saillances détectées pour la reconnaissance de chaque niveau. Par extension, DeepFLE est donc une implémentation des fonctionnalité d'Hyperdeep dédiée au FLE

Interface

Figure 8.13 – DeepFLE : `http://deeptext.unice.fr/FLE`

DeepFLE permet d'évaluer en quelques secondes le niveau d'un texte en français et de visualiser les résultats de l'analyse. Il suffit de copier/coller un texte dans la fenêtre « entrez votre texte » et de cliquer sur « détection du niveau » (figure 8.13). Les utilisateurs peuvent aussi afficher un exemple d'analyse d'un texte prédéfini en cliquant sur « voir un exemple ».

La figure 8.14 illustre l'interface de la plateforme lorsqu'on demande la détection du niveau d'un texte. La prédiction du niveau est affichée aussi bien sous forme de diagramme type radar que de score de reconnaissance. Dans l'exemple de la figure 8.14 le système reconnaît 35,29% de passages-clés de niveau C2 ; 23,53% C1 ; 17,65% B1 ; 17,65% B2 et 5,88% A2. En d'autres mots, le système est capable de détecter les marqueurs spécifiques des différents niveaux du CECRL dans un texte, ce qui est particulièrement utile pour des textes longs de niveaux B et C. De plus, cette analyse permet de décrire l'organisation enchâssée des niveaux.

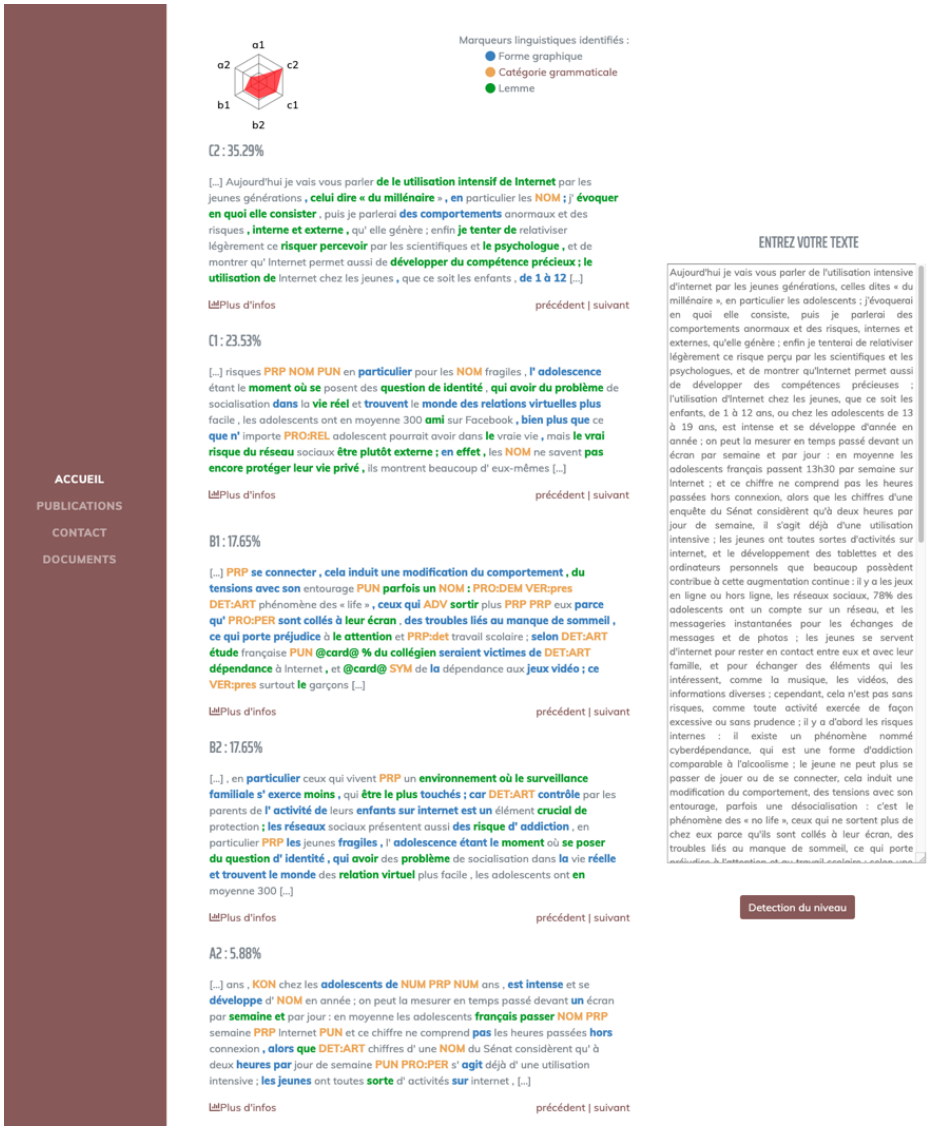


FIGURE 8.14 – Affichage des résultats sur la plateforme DeepFLE

L'approche méthodologique qui croise didactique du FLE, deep learning et analyse des données textuelles conduit donc à la création du premier outil capable d'évaluer, classer et décrire le niveau d'un texte oral en français selon les échelles du CECRL. Un outil en accès libre et destiné aux acteurs du FLE : didacticiens, concepteurs de méthodes, enseignants et également apprenants. Ces premiers résultats ainsi que l'évolution de la performance du deep learning et tout particulièrement du **TDS** pondéré nous amènent à poursuivre nos recherches afin d'optimiser l'analyse prédictive et descriptive des niveaux ainsi que les fonctionnalités de la plateforme DeepFLE.

Chapitre 9

Conclusion

En articulant deep learning et ADT, nous avons cherché à mettre en avant 3 tensions méthodologiques fondamentales dont la communauté ADT peut tirer profit.

— Croiser l'urne (statistique) et le réseau (de neurones), croiser la tokenisation et la convolution, et, d'une autre manière, croiser l'occurrence et la co-occurrence. Le schéma d'urne reste d'une efficacité incontestable pour traiter un corpus textuel contrastif. Cependant, la convolution permet d'ajouter une approche co-textualisée, prenant le mot en contexte, prenant en compte la linérarité du texte ou sa réticularité. La clef de voûte de ces deux approches croisées est sans doute la cooccurrence (les unités du texte dans leur éco-système) comme nous l'avions affirmé, après d'autres, aux JADTs 2016 (Vanni et Mittmann 2016).

— Croiser la classification et la description. L'IA est performante pour classer des textes. L'ADT reste essentiel pour décrire (chiffrer) les régularités et les écarts linguistiques. Le **wTDS** permet d'expliquer la prédiction du réseau de neurones. En d'autres termes, il permet d'ouvrir la boite noire du deep learning et, dans cette ouverture, l'ADT (la statistique) permet de vérifier la pertinence des unités linguistiques exhumées des couches cachées du système.

— Croiser enfin l'axe paradigmatique et l'axe syntagmatique. La langue et les textes sont faits de sélection et de combinaison. Les traitements informatiques et statistiques ADT se sont depuis plusieurs décennies surtout consacrés à repérer les sélections des locuteurs (les mots les plus fréquents, les mots statistiquement préférés, la distance intertextuelle au regard des mots partagés). L'IA (modèle convolutionnel) est sensible à l'axe syntagmatique (les mots dans la fenêtre, dans leurs enchainements ou leurs combinaisons, c'est-à-dire dans leur singularité cotextuelle).

Gageons qu'articulés ensemble les deux approches rendent compte de l'objet textuel dans toute sa complexité.

Le *deep learning* est donc une étape nouvelle dans l'analyse des textes assistée par la machine. Si l'intelligence reste très artificielle, elle n'en demeure pas moins aussi descriptive

qu'une suite d'algorithmes de calculs mathématiques plus ou moins complexes. Il n'y a pas de boite noire pour ceux qui conçoivent leur modèle dans le but d'être interprétable. Le chemin reste encore long pour exploiter les possibilités offertes par ces outils. Les réseaux de neurones profonds remis au gout du jour récemment par la puissance de calcul de nos machines doivent encore trouver leur chemin dans une linguistique textuelle descriptive dominée par des approches plus classiques.

9.1. Perspectives

Nous ne pouvions clore ces travaux sans ajouter quelques remarques sur les réseaux récurrents. Il s'agit là d'une autre architecture de réseau de neurones possible. Certains l'utilisent déjà sur les textes pour modéliser la langue, faire de la traduction automatique ou encore implémenter des systèmes de questions/réponses automatiques (*chatbot*) ou de génération de texte, mais l'écart de performance en classification de texte avec les réseaux convolutionnels n'est pas encore démontré. Il existe des modèles hybrides qui veulent joindre le meilleur des deux méthodes, les scores annoncés sont parfois supérieurs sur certaines tâches, mais leur interprétabilité devient vite un casse-tête qui dissuade de les utiliser à des fins descriptives.

Pourtant leur architecture récurrente propose quelque chose de précieux pour la linguistique textuelle traditionnelle : ces réseaux prennent en compte la temporalité du texte. Le fait qu'un mot soit interprété par le modèle dans un contexte qui garde en mémoire tous les mots qui le précédent fait espérer des effets d'amorçages sémantiques complexes qui pourraient dès lors être observés. Contrairement à la convolution qui associe chaque mot à son contexte proche (fenêtre de convolution), les réseaux récurrents gardent en mémoire l'ensemble du texte, mot après mot dans une série temporelle.

Malheureusement les effets de pertes de mémoire connus de ces réseaux (problème du vanishing gradient), même compensés par les modèles de type *Long Short Term Memory* (LSTM), ne permettent jusqu'à présent pas de mettre au jour des exemples de détection de motifs linguistiques aussi complexes que ceux observés avec la convolution. Tout au plus, une cooccurrence voire une polycooccurrence qui peut être observée par l'usage de couches supplémentaires dite d'*attention* qui poussent le modèle à accentuer l'activation des neurones sur certains mots.

Pour tirer parti de ces réseaux, notre approche de l'analyse des textes doit alors peut-être être modifiée. L'hypothèse contrastive des corpus est certainement une des caractéristiques qui troublent les réseaux récurrents. En effet, leur efficacité a surtout été révélée dans des tâches autres que la classification de texte. Il semble qu'un long corpus homogène d'une seule classe soit préférable pour modéliser des règles linguistiques robustes. Nous devrions nous concentrer alors sur une classe en particulier, une métadonnée, et rassembler un volume de données suffisant pour créer un modèle de génération automatique de texte et espérer voir apparaître des règles linguistiques qui lui sont propres. Il reste le problème de la quantité de données nécessaire, il semble que ce type de réseau soit malheureusement encore plus gourmand en termes de taille que les réseaux convolutionnels traités jusqu'ici.

9.2. Conclusion générale

Comme pour toute nouvelle méthode, la clarté et la qualité de la visualisation des résultats sont souvent responsables du succès ou de l'approbation de la méthode par la communauté. Nous avons vu que la pondération du **TDS** et l'ajout de la lemmatisation et de l'étiquetage morpho-syntaxique des textes rendent les résultats suffisamment explicites pour qu'ils soient utilisés par un linguiste et soumis à son interprétation. Cette méthode prend donc place aujourd'hui dans la plateforme Hyperbase Web qui propose une passerelle logique entre statistique et deep learning et articule ces résultats autour d'une gestion souple des corpus, de la statistique et des réseaux de neurones profonds.

Par cet outil et cette méthode, notre contribution principale est donc une ouverture de la boite noire du deep learning pour les linguistes, qui trouvent en sortie de traitement de leur corpus, des passages-clés et des marqueurs linguistiques caractéristiques de chaque texte, simples tokens ou motifs plus profonds. La déconvolution et le **wTDS** permettent de proposer des parcours interprétatifs d'un genre nouveau, qui rendent compte des saillances multidimensionnelles (formes graphiques, POS, lemmes) apprises et reconnues par le deep learning.

Appendices

Annexe A

Articles en anglais

Text Deconvolution Saliency (TDS) : a deep tool box for linguistic analysis

L. Vanni[1], M. Ducoffe[2], D. Mayaffre[1], F. Precioso[2]
D. Longrée[3], V. Elango[2], N. Santos[2], J. Gonzalez[2], L. Galdo[2], C. Aguilar[1]
[1] Univ. Nice Sophia Antipolis - BCL, UMR UNS-CNRS 7320 - France
[2] Univ. Nice Sophia Antipolis - I3S, UMR UNS-CNRS 7271 - France
[3] Univ. Liège - L.A.S.L.A - Bélgique
laurent.vanni@unice.fr

Abstract

In this paper, we propose a new strategy, called Text Deconvolution Saliency (TDS), to visualize linguistic information detected by a CNN for text classification. We extend Deconvolution Networks to text in order to present a new perspective on text analysis to the linguistic community. We empirically demonstrated the efficiency of our Text Deconvolution Saliency on corpora from three different languages: English, French, and Latin. For every tested dataset, our Text Deconvolution Saliency automatically encodes complex linguistic patterns based on co-occurrences and possibly on grammatical and syntax analysis.

1 Introduction

As in many other fields of data analysis, Natural Language Processing (NLP) has been strongly impacted by the recent advances in Machine Learning, more particularly with the emergence of Deep Learning techniques. These techniques outperform all other state-of-the-art approaches on a wide range of NLP tasks and so they have been quickly and intensively used in industrial systems. Such systems rely on end-to-end training on large amounts of data, making no prior assumptions about linguistic structure and focusing on stastically frequent patterns. Thus, they somehow step away from computational linguistics as they learn implicit linguistic information automatically without aiming at explaining or even exhibiting classic linguistic structures underlying the decision.

This is the question we raise in this article and that we intend to address by exhibiting classic linguistic patterns which are indeed exploited implictly in deep architectures to lead to higher performances. Do neural networks make use of co-occurrences and other standard features, considered in traditional Textual Data Analysis (TDA) (Textual Mining)? Do they also rely on complementary linguistic structure which is invisible to traditional techniques? If so, projecting neural networks features back onto the input space would highlight new linguistic structures and would lead to improving the analysis of a corpus and a better understanding on where the power of the Deep Learning techniques comes from.

Our hypothesis is that Deep Learning is sensitive to the linguistic units on which the computation of the key statistical sentences is based as well as to phenomena other than frequency and complex linguistic observables. The TDA has more difficulty taking such elements into account – such as linguistic linguistic patterns. Our contribution confronts Textual Data Analysis and Convolutional Neural Networks for text analysis. We take advantage of deconvolution networks for image analysis in order to present a new perspective on text analysis to the linguistic community that we call Text Deconvolution Saliency (TDS). Our deconvolution saliency corresponds to the sum over the word embedding of the deconvolution projection of a given feature map. Such a score provides a heat-map of words in a sentence that highlights the pattern relevant for the classification decision. We examine z-test (see section 4.2) and TDS for three languages: English, French and Latin. For all our datasets, TDS highlights new linguistic observables, invisible with z-test alone.

2 Related work

Convolutional Neural Networks (CNNs) are widely used in the computer vision community for a wide panel of tasks: ranging from image classification, object detection to semantic segmentation.

It is a bottom-up approach where we applied an input image, stacked layers of convolutions, non-linearities and sub-sampling.

Encouraged by the success for vision tasks, researchers applied CNNs to text-related problems Kalchbrenner et al. (2014); Kim (2014). The use of CNNs for sentence modeling traces back to Collobert and Weston (2008). Collobert adapted CNNs for various NLP problems including Part-of-Speech tagging, chunking, Named Entity Recognition and semantic labeling. CNNs for NLP work as an analogy between an image and a text representation. Indeed each word is embedded in a vector representation, then several words build a matrix (concatenation of the vectors).

We first discuss our choice of architectures. If Recurrent Neural Networks (*mostly GRU and LSTM*) are known to perform well on a broad range of tasks for text, recent comparisons have confirmed the advantage of CNNs over RNNs when the task at hand is essentially a keyphrase recognition task Yin et al. (2017).

In Textual Mining, we aim at highlighting linguistics patterns in order to analyze their constrast: specificities and similarities in a corpus Feldman, R., and J. Sanger (2007); L. Lebart, A. Salem and L. Berry (1998). It mostly relies on frequential based methods such as z-test. However, such existing methods have so far encountered difficulties in underlining more challenging linguistic knowledge, which up to now have not been empirically observed as for instance syntactical motifs Mellet and Longrée (2009).

In that context, supervised classification, especially CNNs, may be exploited for corpus analysis. Indeed, CNN learns automatically parameters to cluster similar instances and drive away instances from different categories. Eventually, their prediction relies on features which inferred specificities and similarities in a corpus. Projecting such features in the word embedding will reveal relevant spots and may automatize the discovery of new linguistic structures as in the previously cited syntactical motifs. Moreover, CNNs hold other advantages for linguistic analysis. They are static architectures that, according to specific settings are more robust to the vanishing gradient problem, and thus can also model long-term dependency in a sentence Dauphin et al. (2017); Wen et al. (2017); Adel and Schütze (2017). Such a property may help to detect structures relying on different parts of a sentence.

All previous works converged to a shared assessment: both CNNs and RNNs provide relevant, but different kinds of information for text classification. However, though several works have studied linguistic structures inherent in RNNs, to our knowledge, none of them have focused on CNNs. A first line of research has extensively studied the interpretability of word embeddings and their semantic representations Ji and Eisenstein (2014). When it comes to deep architectures, Karpathy et al. Karpathy et al. (2015) used LSTMs on character level language as a testbed. They demonstrate the existence of long-range dependencies on real word data. Their analysis is based on gate activation statistics and is thus global. On another side, Li et al. Li et al. (2015) provided new visualization tools for recurrent models. They use decoders, t-SNE and first derivative saliency, in order to shed light on how neural models work. Our perspective differs from their line of research, as we do not intend to explain how CNNs work on textual data, but rather use their features to provide complementary information for linguistic analysis.

Although the usage of RNNs is more common, there are various visualization tools for CNNs analysis, inspired by the computer vision field. Such works may help us to highlight the linguistic features learned by a CNN. Consequently, our method takes inspiration from those works. Visualization models in computer vision mainly consist in inverting hidden layers in order to spot active regions or features that are relevant to the classification decision. One can either train a decoder network or use backpropagation on the input instance to highlight its most relevant features. While those methods may hold accurate information in their input recovery, they have two main drawbacks: i) they are computationally expensive: the first method requires training a model for each latent representation, and the second relies on backpropagation for each submitted sentence. ii) they are highly hyperparameter dependent and may require some finetuning depending on the task at hand. On the other hand, Deconvolution Networks, proposed by Zeiler et al Zeiler and Fergus (2014), provide an off-the-shelf method to project a feature map in the input space. It consists in inverting each convolutional layer iteratively, back to the input space. The inverse of a discrete convolution is computationally challenging. In re-

sponse, a coarse approximation may be employed which consists of inverting channels and filter weights in a convolutional layer and then transposing their kernel matrix. More details of the deconvolution heuristic are provided in section 3. Deconvolution has several advantages. First, it induces minimal computational requirements compared to previous visualization methods. Also, it has been used with success for semantic segmentation on images: in Noh et al. (2015); Noh et al demonstrate the efficiency of deconvolution networks to predict segmentation masks to identify pixel-wise class labels. Thus deconvolution is able to localize meaningful structure in the input space.

3 Model

3.1 CNN for Text Classification

We propose a deep neural model to capture linguistics patterns in text. This model is based on Convolutional Neural Networks with an embedding layer for word representations, one convolutional with pooling layer and non-linearities. Finally we have two fully-connected layers. The final output size corresponds to the number of classes. The model is trained by cross-entropy with an Adam optimizer. Figure 1 shows the global structure of our architecture. The input is a sequence of words $w_1, w_2 ... w_n$ and the output contains class probabilities (for text classification).

The embedding is built on top of a Word2Vec architecture, here we consider a Skip-gram model. This embedding is also finetuned by the model to to increase the accuracy. Notice that we do not use lemmatisation, as in Collobert and Weston (2008), thus the linguistic material which is automatically detected does not rely on any prior assumptions about the part of speech.

In computer vision, we consider images as 2-dimensional isotropic signals. A text representation may also be considered as a matrix: each word is embedded in a feature vector and their concatenation builds a matrix. However, we cannot assume both dimensions the sequence of words and their embedding representation are isotropic. Thus the filters of CNNs for text typically differ from their counterparts designed for images. Consequently in text, the width of the filter is usually equal to the dimension of the embedding, as illustrated with the red, yellow, blue and green filters in figure 1

Using CNNs has another advantage in our context: due to the convolution operators involved, they can be easily parallelized and may also be easily used by the CPU, which is a practical solution for avoiding the use of GPUs at test time.

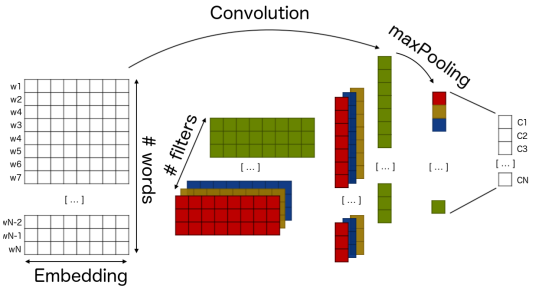


Figure 1: CNN for Text Classification

3.2 Deconvolution

Extending Deconvolution Networks for text is not straightforward. Usually, in computer vision, the deconvolution is represented by a convolution whose weights depends on the filters of the CNN: we invert the weights of the channels and the filters and then transpose each kernel matrix. When considering deconvolution for text, transposing the kernel matrices is not realistic since we are dealing with nonisotropic dimensions - the word sequences and the filter dimension. Eventually, the kernel matrix is not transposed.

Another drawback concerns the dimension of the feature map. Here feature map means the output of the convolution before applying max pooling. Its shape is actually the tuple *(# words, # filters)*. Because the filters' width (red, yellow, blue and green in fig 1) matches the embedding dimension, the feature maps cannot contain this information. To project the feature map in the embedding space, we need to convolve our feature map with the kernel matrices. To this aim, we upsample the feature map to obtain a 3-dimensional sample of size *(# words, embedding dimension, # filters)*.

To analyze the relevance of a word in a sentence, we only keep one value per word which corresponds to the sum along the embedding axis of the output of the deconvolution. We call this sum Text Deconvolution Saliency (TDS).

For the sake of consistency, we sum up our method in figure 2

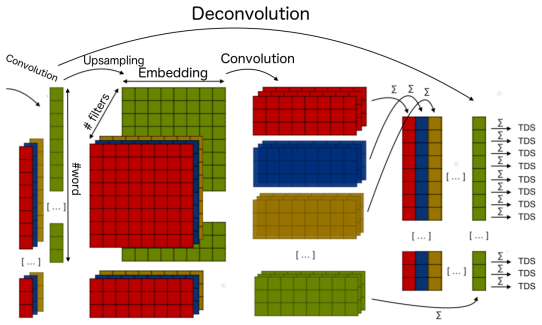


Figure 2: Textual Deconvolution Saliency (TDS)

Eventually, every word in a sentence has a unique TDS score whose value is related to the others. In the next section, we analyze the relevance of TDS. We thoroughly demonstrate empirically, that the TDS encodes complex linguistic patterns based on co-occurrences and possibly also on grammatical and syntaxic analysis.

4 Experiments

4.1 Datasets

In order to understand what the linguistic markers found by the convolutional neural network approach are, we conducted several tests on different languages and our model seems to get the same behavior in all of them. In order to perform all the linguistic statistical tests, we used our own simple linguistic toolbox Hyperbase, which allows the creation of databases from textual corpus, the analysis and the calculations such as z-test, co-occurrences, PCA, K-Means distance,... We use it to evaluate TDS against z-test scoring. We compel our analysis by only presenting cases on which z-test fail while TDS does not. Indeed TDS captures z-test, as we did not find any sentence on which z-test succeeds while TDS fails. Red words in the studied examples are the highest TDS.

The first dataset we used for our experiments is the well known IMDB movie review corpus for sentiment classification. It consists of 25,000 reviews labeled by positive or negative sentiment with around 230,000 words.

The second dataset targets French political discourses. It is a corpus of 2.5 millions of words of French Presidents from 1958 (with De Gaulle, the first President of the Fifth Republic) to 2018 with the first speeches by Macron. In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. The training task is to recognize each french president.

The last dataset we used is based on Latin. We assembled a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin. As with the French dataset, the learning task here is to be able to predict each author according to new sequences of words. The next example is an excerpt of chapter 26 of the 23th book of Livy:

> *[...] tutus tenebat se quoad multum ac diu obtestanti quattuor milia peditum et quingenti equites in supplementum missi ex Africa sunt . tum refecta tandem spe* ***castra propius hostem*** *mouit classem que et ipse instrui parari que iubet ad insulas maritimam que oram tutandam . in* ***ipso impetu*** *mouendarum de [...]*

4.2 Z-test Versus Text Deconvolution Saliency

Z-test is one of the standard metrics used in linguistic statistics, in particular to measure the occurrences of word collocations Manning and Schütze (1999). Indeed, the z-test provides a statistical score of the co-occurrence of a sequence of words to appear more frequently than any other sequence of words of the same length. This score results from the comparison between the frequency of the observerd word sequence with the frequency expected in the case of a "Normal" distribution. In the context of constrative corpus analysis, this same calculation applied to single words can readily provide, for example, the most specific vocabulary of a given author. The highest z-test are the most specific words of this given author in this case. This is a simple but strong method for analyzing features of text. It can also be used to classify word sentences according to the global z-test (sum of the scores) of all the words in the given sentence. We can thus use this global z-test as a very simple metric for authorship classification. The resulting authorship of a given sentence is for instance given by the author corresponding to the highest global z-test on that sentence compared to all other global z-test obtained by summing up the z-test of each word of the same sentence but with the vocabulary specificity of another author. The mean accuracy of assigning the right author to the right sentence, in our data set, is around 87%, which confirms that z-test is indeed meaningful for

	z-test	Deep Learning
Latin	84%	93%
French	89%	91%
English	90%	97%

Table 1: Test accuray with z-test and Deep Learning

contrast pattern analysis. On the other hand, most of the time CNN reaches an accuracy greater than 90% for text classification (as shown in Table 1).

This means that the CNN approaches can learn also on their own some of the linguistic specificities useful in discriminating text categories. Previous works on image classification have highlighted the key role of convolutional layers which learn different level of abstractions of the data to make classification easier.

The question is: what is the nature of the abstraction on text?

We show in this article that CNN approach detects automatically words with high z-test but obviously this is not the only linguistic structure detected.

To make the two values comparable, we normalize them. The values can be either positive or negative. And we distinguish between two thresholds[1] for the z-test: over 2 a word is considered as specific and over 5 it is strongly specific (and the oposite with negative values). For the TDS it is just a matter of activation strength.

The Figure 3 shows us a comparison between z-test and TDS on a sentence extracted from our Latin corpora (Livy Book XXIII Chap. 26). This sentence is an example of specific words used by Livy[2]. As we can see, when the z-test is the highest, the TDS is also the highest and the TDS values are high also for the neighbor words (for example around the word *castra*). However, this is not always the case: for example small words as *que* or *et* are also high in z-test but they do not impact the network at the same level. We can see also on Figure 3 that words like *tenebat*, *multum* or *propius* are totally uncorrelated. The Pearson correlation coefficient[3] tells us that in this sentence there is no linear correlation between z-test and TDS (with a Pearson of 0.38). This example is one of the most correlated examples of our dataset, thus CNN seems to learn more than a simple z-test.

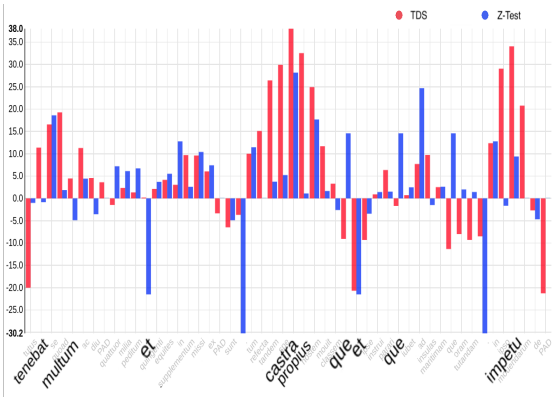


Figure 3: z-test versus Text Deconvolution Saliency (TDS) - Example on Livy Book XXIII Chap. 26

4.3 Dataset: English

For English, we used the IMDB movie review corpus for sentiment classification. With the default methods, we can easily show the specific vocabulary of each class (positive/negative), according to the z-test. There are for example the words *too*, *bad*, *no* or *boring* as most indicitive of negative sentiment, and the words *and*, *performance*, *powerful* or *best* for positive. Is it enough to detect automatically if a new review is positive or not? Let's see an example excerpted from a review from December 2017 (not in the training set) on the last American blockbuster:

> *[...]* ***i enjoyed three moments*** *in the film in total ,* ***and if i am being honest and*** *the person* ***next to me fell asleep*** *in the middle and started snoring during the slow space chasescenes .* ***the story failed to*** *draw me in and entertain* ***me the way*** *[...]*

In general the z-test is sufficient to predict the class of this kind of comment. But in this case, the CNN seems to do better, but why?

[1] The z-test can be approximated by a normal distribution. The score we obtain by the z-test is the standard deviation. A low standard deviation indicates that the data points tend to be close to the mean (the expected value). Over 2 this score means there is less than 2% of chance to have this distribution. Over 5 it's less than 0.1%.

[2] Titus Livius Patavinus – (64 or 59 BC - AD 12 or 17) – was a Roman historian.

[3] Pearson correlation coefficient measures the linear relationship between two datasets. It has a value between $+1$ and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative

If we sum all the z-test (for negative and positive), the positive class obtains a greater score than the negative. The words *film*, *and*, *honest* and *entertain* – with scores 5.38, 12.23, 4 and 2.4 – make this example positive. CNN has activated different parts of this sentence (as we show in bold/red in the example). If we take the sub-sequence *and if i am being honest and*, there are two occurences of *and* but the first one is followed by *if* and our toolbox gives us 0.84 for *and if* as a negative class. This is far from the 12.23 in the positive. And if we go further, we can do a co-occurrence analysis on *and if* on the training set. As we see with our co-occurrence analysis[4] (Figure 4), *honest* is among the most specific adjectivals[5] associated with *and if*. Exactly what we found in our example.

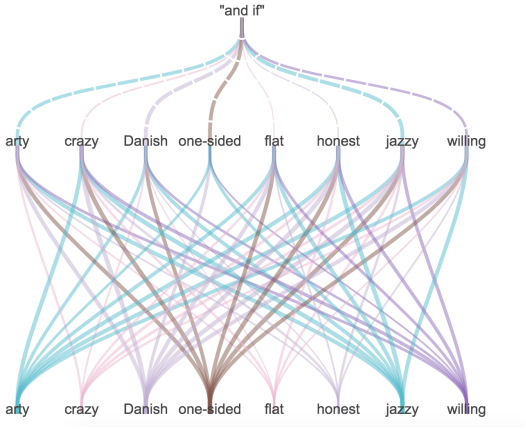


Figure 4: co-occurrences analysis of *and if* (Hyperbase)

In addition, we have the same behavior with the verb *fall*. There is the word *asleep* next to it. *Asleep* alone is not really specific of negative review (z-test of 1.13). But the association of both words become highly specific of negative sentences (see the co-occurrences analysis - Figure 5).

The Text Deconvolution Saliency here confirms

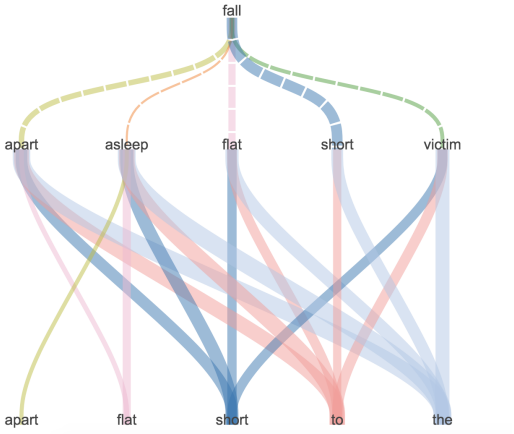


Figure 5: co-occurrences analysis of *fall* (Hyperbase)

that the CNN seems to focus not only on high z-test but on more complex patterns and maybe detects the lemma or the part of speech linked to each word. We will see now that these observations are still valid for other languages and can even be generalized between different TDS.

4.4 Dataset: French

In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. In this speech, the CNN primarily recognizes Macron (the training task was to be able to predict the correct President). To achieve this task the CNN seems to succeed in finding really complex patterns specific to Macron. For example in this sequence:

> *[...] notre pays* ***advienne à*** *l'école pour nos enfants, au travail pour l' ensemble de* ***nos concitoyens*** *pour le climat pour le quotidien de chacune et chacun d' entre vous .* ***Ces transformations profondes*** *ont commencé et se* ***poursuivront*** *avec la même force le même rythme la même intensité [...]*

The z-test gives a result statistically closer to De Gaulle than to Macron. The error in the statistical attribution can be explained by a Gaullist phraseology and the multiplication of linguistic markers strongly indexed with De Gaulle: De Gaulle had the specificity of making long and literary sentences articulated around co-ordination conjunctions as in *et* (z-test = 28 for de Gaulle, two occurrences in the excerpt). His speech was also

[4]Those figures shows the major co-occurrences for a given word (or lemma or PartOfSpeech). There two layers of co-occurrences, the first one (on top) show the direct co-occurrence and the second (on bottom) show a second level of co-occurrence. This level is given by the context of two words (taken together). The colors and the dotted lines are only used to make it more readable (dotted lines are used for the first level). The width of each line is related to the z-test score (more the z-test is big, more the line is wide).

[5]With our toolbox, we can focus on different part of speech.

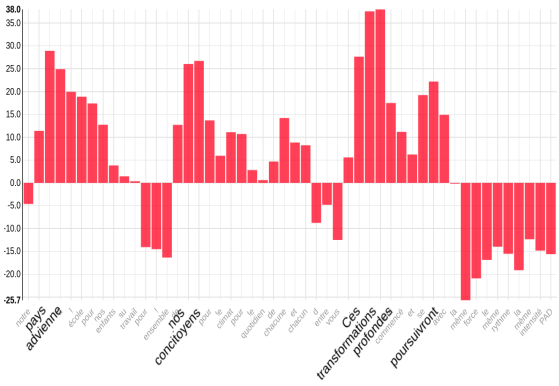


Figure 6: Deconvolution on Macron speech.

more conceptual than average, and this resulted in an over-use of the articles defined *le*, *la*, *l´*, *les*) very numerous in the excerpt (7 occurrences); especially in the feminine singular (*la république*, *la liberté*, *la nation*, *la guerre*, etc., here we have *la même force*, *la même intensité*.

The best results given by the CNN may be surprising for a linguist but match perfectly with what is known about the sociolinguistics of Macron's dynamic kind of speeches.

The part of the excerpt, which impacts most the CNN classification, is related to the nominal syntagm *transformations profondes*. Taken separately, neither of the phrase's two words are very Macronian from a statistical point of view (*transformations* = 1.9 *profondes* = 2.9). Better, the syntagm itself does not appear in the President's learning corpus (0 occurrence). However, it can be seen that the co-occurrence of *transformation* and *profondes* amounts to 4.81 at Macron: so it is not the occurrence of one word alone, or the other, which is Macronian but the simultaneous appearance of both in the same window. The second and complementary most impacting part of the excerpt thus is related to the two verbs *advienne* and *poursuivront*. From a semantic point of view, the two verbs perfectly contribute, after the phrase *transformations profondes*, to give the necessary dynamic to a discourse that advocates change. However it is the verb tenses (carried by the morphology of the verbs) that appear to be the determining factor in the analysis. The calculation of the grammatical codes co-occurring with the word *transformations* thus indicates that the verbs in the subjunctive and the verbs in the future (and also the nouns) are the privileged codes for Macron (Figure 7).

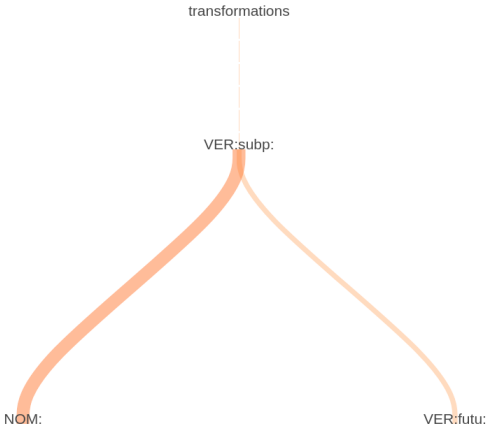


Figure 7: Main part-of-speech co-occurrences for *transformations* (Hyperbase)

More precisely the algorithm indicates that, for Macron, when *transformation* is associated with a verb in the subjunctive (here *advienne*), then there is usually a verb in the future co-present (here *poursuivront*). *transformations profondes*, *advienne* to the subjunctive, *poursuivront* to the future: all these elements together form a speech promising action, from the mouth of a young and dynamic President. Finally, the graph indicates that *transformations* is especially associated with nouns in the President's speeches: in an extraordinary concentration, the excerpt lists 11 (*pays*, *école*, *enfants*, *travail*, *concitoyens*, *climat*, *quotidien*, *transformations*, *force*, *rythme*, *intensité*).

4.5 Dataset: Latin

As with the French dataset, the learning task here is to be able to predict the identity of each author from a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin.

The statistics here identify this sentence as Caesar[6] but Livy is not far off. As historians, Caesar and Livy share a number of specific words: for example tool words like *se* (reflexive pronoun) or *que* (a coordinator) and prepositions like *in*, *ad*, *ex*, *of*. There are also nouns like *equites* (cavalry) or *castra* (fortified camp).

The attribution of the sentence to Caesar cannot only rely only on z-test: *que* or *in* or *castra*, with differences thereof equivalent or inferior to Livy.

[6]Gaius Julius Caesar, 100 BC - 44 BC, usually called Julius Caesar, was a Roman politician and general and a notable author of Latin prose.

On the other hand, the differences of *se*, *ex*, are greater, as is that of *equites*. Two very Caesarian terms undoubtedly make the difference *iubet* (he orders) and *milia* (thousands).

The greater score of *quattuor* (four), *castra*, *hostem* (the enemy), *impetu* (the assault) in Livy are not enough to switch the attribution to this author.

On the other hand, CNN activates several zones appearing at the beginning of sentences and corresponding to coherent syntactic structures (for Livy) – *Tandem reflexes spe castra propius hostem mouit* (then, hope having finally returned, he moved the camp closer to the camp of the enemy) – despite the fact that *castra* in *hostem mouit* is attested only by Tacitus[7].

There are also *in ipso metu* (in fear itself), while *in* followed by *metu* is counted one time with Caesar and one time also with Quinte-Curce[8].

More complex structures are possibly also detected by the CNN: the structure *tum* + participates Ablative Absolute (*tum refecta*) is more characteristic of Livy (z-test 3.3 with 8 occurrences) than of Caesar (z-test 1.7 with 3 occurrences), even if it is even more specific of Tacitus (z-test 4.2 with 10 occurrences).

Finally and more likely, the co-occurrence between *castra*, *hostem* and *impetu* may have played a major role: Figure 8

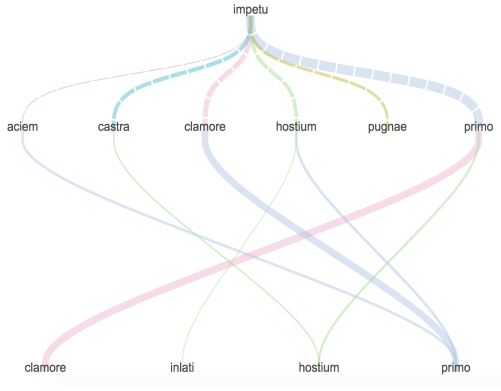


Figure 8: Specific co-occurrences between *impetu* and *castra* (Hyperbase)

With Livy, *impetu* appears as a co-occurrent with the lemmas *hostis* (z-test 9.42) and *castra* (z-test 6.75), while *hostis* only has a gap of 3.41 in Caesar and that *castra* does not appear in the list of co-occurrents.

For *castra*, the first co-occurent for Livy is *hostis* (z-test 22.72), before *castra* (z-test 10.18), *ad* (z-test 10.85), *in* (z-test 8.21), *impetus* (z-test 7.35), *que* (z-test 5.86) while in Caesar, *impetus* does not appear and the scores of all other lemmas are lower except *castra* (z-test 15.15), *hostis* (8), *ad* (10,35), *in* (5,17), *que* (4.79).

Thus, our results suggest that CNNs manage to account for specificity, phrase structure, and co-occurence networks...

5 Conclusion

In a nutshell, Text Deconvolution Saliency is efficient on a wide range of corpora. By crossing statistical approaches with neural networks, we propose a new strategy for automatically detecting complex linguistic observables, which up to now hardly detectable by frequency-based methods. Recall that the linguistic matter and the topology recovered by our TDS cannot return to chance: the zones of activation make it possible to obtain recognition rates of more than 91% on the French political speech and 93% on the Latin corpus; both rates equivalent to or higher than the rates obtained by the statistical calculation of the key passages. Improving the model and understanding all the mathematical and linguistic outcomes remains an import goal. In future work, we intend to thoroughly study the impact of TDS given morphosyntactic information.

[7]Publius (or Gaius) Cornelius Tacitus, 56 BC - 120 BC, was a senator and a historian of the Roman Empire.

[8]Quintus Curtius Rufus was a Roman historian, probably of the 1st century, his only known and only surviving work being "Histories of Alexander the Great"

References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1723–1729.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941.

Feldman, R., and J. Sanger. 2007. *The Text Mining Handbook. Advanced Approaches in Analyzing Un-*

structured Data. New York: Cambridge University Press.

Hyperbase. Web based toolbox for linguistics analysis. http://hyperbase.unice.fr.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 655–665.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

L. Lebart, A. Salem and L. Berry. 1998. *Exploring Textual Data*. Ed. Springer.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

S. Mellet and D. Longrée. 2009. Syntactical motifs and textual structures. In *Belgian Journal of Linguistics 23*, pages 161–173.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

A Supplemental Material

In order to make our experiments reproductible, we detail here all the hyperparameters used in our architecture. The neural network is written in python with the library Keras (an tensorflow as backend).

The embedding uses a Word2Vec implementation given by the gensim Library. Here we use the SkipGram model with a window size of 10 words and output vectors of 128 values (embedding dimension).

The textual datas are tokenized by a homemade tokensizer (which work on English, Latin and French). The corpus is splited into 50 length sequence of words (punctuation is keeped) and each word is converted inta an uniq vectore of 128 value.

The first layer of our model takes the text sequence (as word vectors) and applies a weight corresponding to our WordToVec values. Those weights are still trainable during model training.

The second layer is the convolution, a Conv2D in Keras with 512 filters of size $3 * 128$ (filtering three words at time), with a Relu activation method. Then, there is the Maxpooling (MaxPooling2D)

(The deconvolution model is identical until here. We replace the rest of the classification model (Dense) by a transposed convolution (Conv2DTranspose).)

The last layers of the model are Dense layers. One hidden layer of 100 neurons with a Relu activation and one final layer of size equal to the number of classes with a softmax activation.

All experiments in this paper share the same architecture and the same hyperparameters, and are trained with a cross-entropy method (with an Adam optimizer) with 90% of the dataset for the training data and 10% for the validation. All the tests in this paper are done with new data not included in the original dataset.

Key passages : from statistics to deep learning

Laurent Vanni[1], Marco Corneli[2], Dominique Longree[3], Damon Mayaffre[4], Frederic Precioso[5]

Abstract This contribution compares statistical analysis and deep-learning approaches to textual data. The extraction of *key passages* using statistics and deep learning is implemented using the Hyperbase software. An evaluation of the underlying calculations is given by using examples from two different languages – French and Latin. Our hypothesis is that deep learning is not only sensitive to word frequency but also to more complex phenomena containing linguistic features that pose problems for statistical approaches. These linguistic patterns, also known as *motives* (Mellet and Longrée, 2009) are essential for highlighting key-passages. If confirmed, this hypothesis would provide us with a better understanding of the deep learning *black box*. Moreover, it would bring new ways of understanding and interpreting texts. Thus, this paper introduces a novel approach to explore the hidden layers of a convolutional neural network, trying to explain which are the relevant linguistic features used by the network to perform the classification task. This explanation attempt is the major contribution of this work. Finally, in order to show the potential of our deep learning approach, when testing it on the two corpora (French and Latin), we compare the obtained linguistic features with those highlighted by a standard text mining technique (z-score computing).

[1] Université Côte d'Azur, CNRS, BCL (UMR 7320), France, e-mail: laurent.vanni@univ-cotedazur.fr.
[2] Université Côte d'Azur, Center of Modeling, Simulmation & Interaction, France, e-mail: Marco.Corneli@univ-cotedazur.fr.
[3] L.A.S.L.A., e-mail: dominique.longree@uliege.be.
Université Côte d'Azur, CNRS, BCL (UMR 7320), France, e-mail: damon.mayaffre@univ-cotedazur.fr.
[5] Université Côte d'Azur, CNRS, I3S (UMR 7271), France, e-mail: Frederic.Precioso@univ-cotedazur.fr.

1 Introduction

Due mainly to technical reasons, until the 1960s textual analysis developed around the "token", which in computer science denotes the string between two white spaces.[1] Since then, practitioners have continued to widen the set of observable features, believing that the complexity of a text cannot be reflected by tokens alone. Thus, although the tokenization, along with computing word specificity[2], are still the preliminary steps in the statistical analysis of textual data, research into wider and more complex phraseological units characterizing the structure of texts has now become the focus of the discipline. Indeed, since 1987, the calculation of the *repeated segment* (Salem, 1987) or n-grams has marked a breakthrough, since significant segments of the text, with no predetermined size, were automatically detected. Further, the automatic and unsupervised detection of *patterns*[3](Mellet and Longrée, 2009; Quiniou et al., 2012; Mellet and Longrée, 2012; Longrée and Mellet, 2013) is a key issue. From this perspective, the present contribution focuses on the notion of *key passages*, which we define in Section 2.1 of the text, as they are implemented in the Hyperbase[4] software. The outlined approach consists of two steps. First, we propose a statistical extraction of key features, with an evaluation of their interpretative relevance on two corpus, a French and a Latin one. Then, a methodological comparison with a deep learning approach is implemented. Thus, by using the Text Deconvolution Saliency (TDS) method (Vanni et al., 2018), we identify within each corpus areas of activation (Section 3.2.2 and observe that, from a linguistic point of view, they constitute relevant *patterns*.

The paper is organized as follows. Section 2 introduces the notion of key passage and reviews a standard statistical approach. Section 3 outlines an original deep learning approach for key passage detection and illustrates how TDS can be used to highlight some key *patterns* in the text. Section 4 concludes the paper.

2 Key-passages and statistics

2.1 Definition

In the following, the word "passage" will be used to denote any text portion *not* necessarily being a sentence of a paragraph. Indeed, the approaches outlined in this

[1] This contribution has been founded by the french government, Agence Nationnale de la Recherche, project Investissement d'Avenir UCA^{JEDI} n° ANR-15-IDEX-01

[2] Also known as z-score, specificity in textual data analysis since (Lafon, 1984) is based on hypergeometric distribution.

[3] Also known as motives: complex linguistic objects with variable and discontinuous spans

[4] The software is used by the UMR Bases, Corpus, Language and was developed in collaboration with the LASLA. A first local version was coded by Etienne Brunet. A recent web development was realized by Laurent Vanni http://hyperbase.unice.fr/.

paper do not have a syntactic model. Thus, strong punctuation delimiting a sentence, although useful, is not needed for our processing. As theorized by Rastier (2007) in an eponymous article, the notion of *passage* refers to the interpretability of a part of a text. A passage is therefore a piece of text judged meaningful enough to characterize the whole text, independently of its size (one word, one phrase, etc.). Based on this idea, we can formally define a key passage as a text portion containing a high number of specificities (Lafon, 1984). Textometric software such as Hyperbase, Dtm-Vic, Iramuteq implement key passage extraction. The more specificities a passage contains, the more remarkable it is considered. The next section illustrates the given definition with two examples.

2.2 Implementations

2.2.1 Unfiltered processing

Let us denote the i-th text portion (henceforth text) by y_i, that can be seen as a vector of length N_i, where N_i is the number of words in that text. The corpus considered for the experiments in Section 3.3 is constituted by texts sharing the same length, therefore $N_i = N$ for all i. The j-th entry of y_j, namely $y_{ij} \in \mathbb{N}$ identifies the j-th token (of the i-th text) via a natural number, corresponding to the ID of the token in a dictionary containing V vocables. The dictionary is obtained by collecting all the different vocables appearing in the corpus. So, for instance, if the first token of the fifth passage is "nation" and the vocable "nation" is the 25-th word of the dictionary, then $y_{51} = 25$. In a standard statistical analysis, when inspecting a text, the specificity indexes of words can be summed in order to obtain a score characterizing the whole text. Moreover, each word has several specificity indexes, corresponding (for instance) to different authors. More formally, let us denote by K the number of different authors. A real matrix Z, with K rows and V columns is introduced such that Z_{kv} is the specificity index of the v-th vocable with respect to the k-th author. Roughly speaking, Z_{kv} measures (in terms of numbers of standard deviations) how much the word v is overused or underused by the k-th author with respect to the average in the whole corpus. Then, a score S_{ik} can be computed for y_i with respect to the k-th author as

$$S_{ik} = \sum_{j=1}^{N} Z_{k y_{ij}}. \tag{1}$$

Note that the specificity indexes in Z can be either positive or negative. A positive index corresponds to a word v that is overused by an author. Thus, that word contributes towards the identification of a passage as being representative of that author. Conversely, a negative index marks a word as underused by the author. Thus, negative indexes contribute to a "non-identification" of the passage. Once the scores S_{ik} have been assigned to all texts for all the authors, Hyperbase sorts the texts from

the more representative (highest S_{ik}) to the least representative (lowest S_{ik}) for each author. In the unfiltered version, the scores S_{ik} are computed as in Eq. (1).

2.2.2 Filtered processing

With our linguistic and statistical backgrounds, the approach described in the previous section can be refined and improved. For instance, only positive specificities – and among them, the strongest ones – might be considered, based on the assumption that an object is better identified by its strengths than by its weaknesses. Thus, modified scores S_{ik} can be computed as

$$S_{ik} = \sum_{j=1}^{N} Z_{ky_{ij}} \mathbf{1}_{[c,+\infty[}(Z_{ky_{ij}}), \tag{2}$$

where c is a positive constant, corresponding to the desired threshold and $\mathbf{1}_A(\cdot)$ denotes the indicator function on a set A.

Additional filters might be considered. For instance, the tool words (conjunctions, determinants) could be discarded. Indeed, they bring the double disadvantage of having very high frequencies (potentially crucial when computing the specificities) and of not being very meaningful, from a semantic/thematic point of view. Finally, the grammatical category can be considered. For example, using only proper and common names can be more efficient than using the entire set of words.

2.3 Examples

Key passage detection via the Hyperbase software is performed on two different data-sets. The first contains the speeches of the French presidents of the 5th republic and the second is a Latin corpus that regroups texts from the principle classical authors. See Table 1 for more details.

dataset	authors	vocab	occurences
French	8	46978	2 738 652
Latin	22	56242	2 035 176

Table 1 French and Latin datasets.

2.3.1 French corpus

With a view to characterizing contemporary French political discourse, the statistical extraction of Emmanuel Macron's key passages (versus his predecessors at the Elysée: de Gaulle, Pompidou, Giscard, Mitterrand, Chirac, Sarkozy, and Holland) represents obvious added value for the linguist and historian. In a corpus of several tens of thousands of passages, the machine selects the following extract:

> *[. . .] l'engagement de l'Europe pour réussir justement cette aventure de l'intelligence artificielle. Donc, vous l'avez compris, je souhaite que la France soit l'un des leaders de cette intelligence artificielle, je souhaite que l'Europe soit l'un des leaders de cette intelligence artificielle [. . .]*

(Macron, the 29th March 2018, speech "AI for Humanity" at the collège de France)

Everything in this passage is typical of Macron's speeches. From an enunciative point of view, *je* / *vous* is a persuasive rhetorical device where the polite but repeated affirmation of the president's personality (*je souhaite*, *je souhaite*) versus people in a position of inferiority (*vous avez compris*). In France, this politico-enunciative posture, marked by the over-use of *je* and of *vous* has been described by commentators as a "Jupiterian" or dominant posture. Especially from a lexical point of view, two major themes of macronism seem to be highlighted by the machine: Europe on the one hand, and post-industrial modernity on the other. Europe has been a theme throughout the French presidential corpus since 1958. But Macron, more than anyone else has used the word *Europe* with such recurrence (Figure 1).

And in this excerpt, as is often the case in Macron's speech, if the President still speaks of *France*, it is indeed *Europe*, twice repeated, that is the subject of the speech. Post-industrial productionism, referred to here as *intelligence artificielle*, represents another facet of Macron's fundamental message since he was elected President: digital modernity, digital sciences, and artificial intelligence represent the economic future. And the terms *réussir* or *leaders* end up characterizing this modern discourse on the *start-up nation*.

2.3.2 Latin corpus

Focusing on the Latin corpus, we consider the following key passage from Julius Caesar, in contrast with many authors contained in the LASLA database:

> *[...] partes Galliae uenire audere quas Caesar possideret neque exercitum sine magno commeatu atque molimento in unum locum contrahereposse sibi autem mirum uideri quid in sua Gallia quam bello uicisset aut Caesari aut omnino populo Romano negotii esset his responsis ad Caesarem relatis iterum ad eum Caesar [...]*

(César, Bellum Gallicum, 1, 34)

This passage from the *Gallic Wars* can indeed be considered as very representative of Caesar's work. There are known proper names (*Galliae*, *Caesar*, *Gallia*) or common names reflecting the military context of the time (*bello*, *commeatu*).

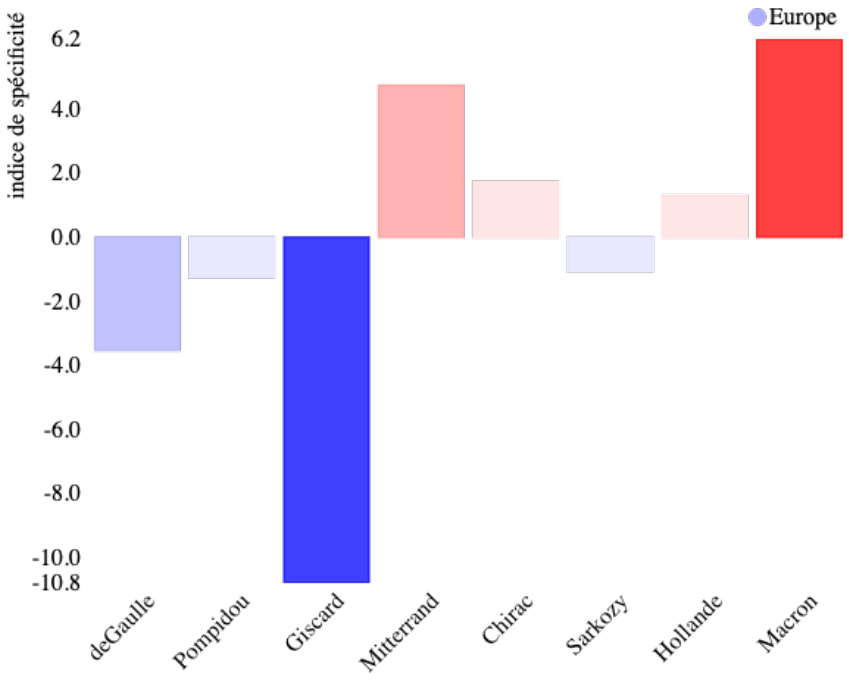


Fig. 1 Distribution of word “Europe" in the french presidential corpus

However, the method does not allow the identification of structures characteristic of Caesar’s language and style, such as a specific participial clause marking the transition between episodes in a negotiation: *His responsis ad Caesarem relatis*, “These answers having been reported to Caesar”.

3 Key passages and deep learning

3.1 Most relevant passages detected by the neural network

In machine learning, the observed data sets are usually split into three parts: a *training* data set (e.g. 80% of all the data), a *validation* data set (e.g. 10 % of all the data) and a *test* data set (e.g. 10 % of all the data). In a classification framework, for instance, the training data-set is used to learn the parameters of a classifier, via minimization of a given loss function. During optimization, the loss function is progressively monitored on both the training and the validation data sets, in order

to avoid *overfitting*. This phenomenon occurs when a model obtains a very high accuracy score on the training data-set but cannot generalize to the rest of the data. On the contrary, a smart classifier is asked to work reasonably well on new, unseen data sets like the test data-set. Thus, a reasonable goal is to reach the highest possible accuracy scores on the *test* data-set. However, for our purpose, which is the detection of the key passages, the description of the training data set is at least as crucial as the performance on the test data set. We stress that we are interested in understanding how the network accomplishes the classification task. We are now in a position to take a first step in that direction based on the following idea. The number of neurons in the last layer of a deep neural network generally coincides with the number of classes, previously denoted by K. Let us denote by[5] $x_i \in \mathbb{R}^K$ the value of that layer for the i-th text. Thus, x_{ik} is the value of the k-th neuron and it is a real number. A softmax activation function is usually applied to x_i in such a way to obtain K probabilities p_{ik} lying in the $K-1$ simplex

$$p_{ik} = \frac{\exp(x_{ik})}{\sum_{j=1}^{K} \exp(x_{ij})}. \tag{3}$$

The highest probability corresponds to the class the observation is assigned to by the network. However, if one entry of x_i is significantly higher than the others, it is mapped to 1 by the softmax transformation and all the other entries are mapped to zero. Assuming that two layers $x_i \neq x_j$ (corresponding to two different texts) both have the k-th component sufficiently high, after applying the softmax transformation it will be impossible to know which of them had the highest activation rate toward cluster k. Thus, we make unconventional use of the trained deep-neural network and observe the activation rate of neurons *before* applying the softmax transformation. Doing that allows us to sort the learning data (texts) based on their activation strengths. Eventually, only the texts which most strongly activate their classes are retained. This simple but efficient method provides us with the most relevant passages in the corpus for each class. Those passages will be referred to as "deep learning key-passages" in what follows.

3.2 Learning linguistic features

Once the the deep learning key passages have been collected, we still need to consider the reason why the neural network treated them as most relevant. What are the linguistic markers learned by the network that contribute to the classification? How to display them? Whereas in standard statistical analysis those markers are the word specificities and they are computed before the classification, in deep learning the textual features are learned during the training process. In order to visualize this type of linguistic material, we must analyze the deep-neural network's hidden layers.

[5] In general, $x \in \mathbb{R}^N$ can be read as "a vector of N real components"

There are several types of neural networks that seem to be sensitive to different markers in the text. In this work we mainly focus on convolutional neural networks (CNNs), usually employed for image classification. Convolution is a powerful tool, invariant with respect to translation and scale, allowing the CNN to detect key features in images. See Krizhevsky et al. (2012) for more details. In order to analyze text data with CNNs, fixed-length input sequences are needed (Ducoffe et al., 2016). Therefore, the whole corpus is split into texts of 50 tokens each.[6]. Then, an embedding on the word-level is adopted to convert words into real vectors of fixed size, being a network hyper-parameter. Several embedding techniques are available, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2017; Joulin et al., 2017). Note that working with textual features the size of words rather than the size of characters is more practical for linguists and leads to results which are easier to interpret. The word vectors are stacked by row in an embedding matrix whose entries are considered parameters to be learned. In our model, the convolutional layer is the most important one and it is expected to capture most of the linguistic features of the training data set. The next section describes this layer in detail.

3.2.1 Convolution

The convolution applied to the text uses filter mechanisms responsible for analyzing each word in its context to extract a salience score for each analyzed context. It is then the combination of all these features that pushes the network to make its decision. This combination can be as simple as a sum of contextual specificities (colocations or repeated segments) or as complex as detecting all the specific co-occurrences or even poly-coccurrences of expressions (Vanni and Mittmann, 2016).

As we can see with Figure 2, the complete network is composed of three parts: embedding, convolution and classification. The last one is a dense network whose last layer is composed of a number of neurons equal to the number of classes to be detected (8 for French and 22 for Latin).

3.2.2 Deconvolution

The idea of deconvolution as presented in (Vanni et al., 2018) is to propose the opposite path. Taking the abstraction made by convolution (the feature map), and then returning to a vector space similar to the input embedding in order to be able to read the linguistic highlights learned by the network. It is a convolution transposed on all the filters that allows this operation to be performed (Figure 3). Thus, we obtain a salience score for each word that represents the weight of each of the linguistic

[6] The number of tokens for each text is one of several hyper-parameters of the network.

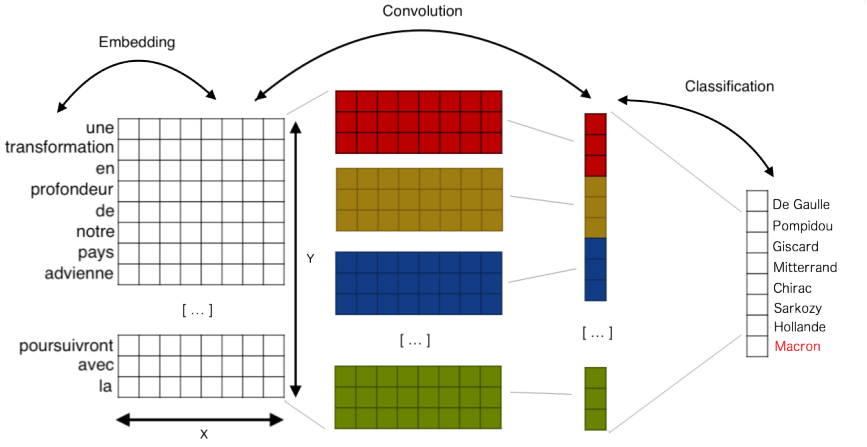


Fig. 2 Convolutional Neural Network (CNN)

elements in the final deep-learning decision. This score is the Text Deconvolution Saliency (TDS).

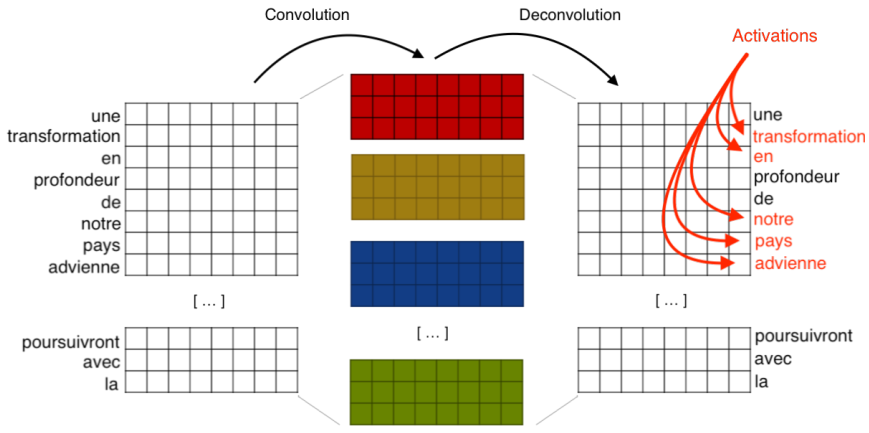


Fig. 3 Deconvolutional Neural Network (DNN)

This is where the notion of passage takes on its full meaning. Each of the words is assigned a certain weight by convolution, but it is essential to understand that this weight is entirely dependent on the context of these words. The convolution filters all have a size that defines the window in which each word will be considered. This weight can therefore differ greatly from one context to another. We can well imagine here that the network might be sensitive to any type of micro-distributional markers (colocaltion, repeated segments, phrases, etc.). But in deep learning the network must be considered as a whole and therefore the layers that follow convolution (MaxPooling, Dense,...) can also largely affect the network's decision. Thus, each linguistic marker is itself considered in the context of the entire word sequence (co-occurrence and poly-cooccurrence of expressions). One last notion that is important to emphasize is that the repetition of examples and the size of the training corpus necessarily pushes the network to also be sensitive to frequency type markers, such as specificities for example.

We can see here that deep learning potentially plays on several levels at the same time to learn and detect strong linguistic markers characteristic of each author. There is no a priori on the object being studied; the word or pattern, the micro-distribution or macro-distribution. These methods can therefore potentially detect all types of linguistic patterns described by (Mellet and Longrée, 2009). Unfortunately, deconvolution does not take into account the rest of the network, so we still lack some elements for automatically observing such patterns. But what we do know now is that each strong activation in a sequence of words visibly points to a prominent passage in the text. These very Rastierian passages can be simple specific words or longer segments with co-occurrences or poly-cooccurrences. This method offers us new linguistic markers that scholars can use today to complement statistical analysis of textual data to provide new avenues to explore in the course of their research.

3.2.3 Filtered processing

As with the statistical approach, our linguistic and statistics background allows us to add certain filters prior to the deep-learning analysis to avoid all kinds of bias in the training corpus and thus refine the textual-features detected by the model. For example in the corpus of French presidents, one-letter words and cardinals have been removed. Indeed, this corpus is strongly constrained by chronology. The dates and the changing punctuation strongly mark each president's speech and have to be removed to maintain a maximum of homogeneity in the corpus.

Some words or expressions are also strongly guided by chronology. For example it is totally implausible to find the name "Trump" in the C. De Gaulle speech. F. Hollande and E. Macron are the only French presidents who talk about the current American president. This information could be considered by deep learning as sufficient to characterize a president but from the point of view of linguistics it is definitely not enough to characterize a political speech. The analysis of word distribution therefore allows us to remove those unique words or expressions which could mask the true key-passages of a text.

The following examples therefore use this filter mechanism to observe the most relevant text features that characterize a text.

3.3 Examples

3.3.1 French dataset

In the presidential corpus, the passage that the AI identifies as characteristic of Macron's speech, gathers remarkable features of the current French president's language, and the deconvolution highlights observable linguistic elements with multiple interpretations:

[...] entreprises de manière légitime, qui permettra aux acteurs européens d'émerger dans un marché loyal et qui permettra aussi de compenser les profondes désorganisations sur l'économie traditionnelle que cette transformation parfois crée. Les grandes plateformes numériques, la protection des données sont au cœur de notre souveraineté [...]

(Macron, the 26th of September 2017, speech about Europe at the Sorbonne)

One of the most interesting linguistic features is the relative pronoun *qui* which is a typical element in Macron's construction of phrases. With this relative pronoun, which drives the speech, the complexity of the phrase is brought to the forefront. The second characteristic revealed is the sequence of two verbs in the future (*permettra* [...] *permettra*) and one verb in the present tense (*sont*): here too it is a form of promise, most often combined with the future, and a pragmatism declined in the present tense that are brought to light. Last but not least, a characteristic lexicon is constructed to form an ambitious, dynamic and entrepreneurial discourse that distinguishes Macron's discourse from that of a De Gaulle, a Pompidou or a Mitterrand: *entreprises*, *acteurs*, *européens*, *marché*, *transformation*, *crée*, *numériques*, *données*.

3.3.2 Latin dataset

Deep learning can also be easily adapted to the subtleties of each language. This is what we will try to demonstrate here by using Latin and the variation that is specific to it. In the next example we focus on the following deep learning key passage from Julius Caesar (the one of most relevant according to the AI).

[...] ut tum accidit . munitionem quam pertinere a castris ad flumen supra demonstrauimus dextri Caesaris cornus cohortes ignorantia loci sunt secutae cum portam quaererent castrorum que eam munitionem esse arbitrarentur . quod cum esset animaduersum coniunctam esse flumini prorutis munitionibus defendente nullo transcenderunt omnis que noster equitatus eas cohortes est [...]

(César, Belum ciuile, 3, 68)

In this excerpt, deep learning recognized an occurrence of a "textual motif" whose pattern has been clearly identified before [conjonction ut or relative pronoun

+ supra/antea + 1ers person of the verbs demonstro/dico/memoro] (Longrée and Mellet, 2013; Longrée et al., 2019). Simple variants of the motif as *ut supra demonstraui* (“as I have explained above") or *quem antea memorauimus* (“whom we mentionned before") were quite easily identified by using standard data mining methods. However, such methods are unable to retrieve a variant of the motif as complex as the one found in the excerpt, *munitionem, quam pertinere a castris ad flumen supra demonstrauimus* (“the entrenchment, which, as we have explained above, extended from the camp to the river") where the relative pronoun is not the complement of the verb as in *quem antea memorauimus*, but the subject of an infinitive clause depending on the verb *demonstrauimus*. Once again, deep learning has shown its efficiency in identifying texts, but also in mining multidimensional structures. In the same passage, deep learning detects another structure which can also be considered as textual motif, *Quod cum esset animaduersum coniunctam esse flumini, prorutis munitionibus defendente nullo* (“And as it had been observed that it was connected with the river, the defenses having been thrown down, no one defending it"). This syntactic motif consists in a *cum* clause followed by two absolute ablatives (Mellet and Longrée, 2009). Moreover this syntactic p includes an occurrence of the multidimensional motif *quod ubi cognitum est* (“when this has been known") made of the coordinating relative *quod* followed by a subordinative conjunction and a passive verb form (Longrée et al., 2019). As Longrée and Mellet (2013) defines the “textual motif” as a multidimensional unit fulfilling characterizing and structuring textual functions, it is not very surprising that deep learning confirms the saliency of such patterns for text identification.

4 Conclusion

4.1 Discussion

Standard statistical analysis and deep learning are not fully disconnected fields. Indeed, the word-frequency based approach, typical of the standard statistical techniques, is also used by deep-learning approaches (Lebart, 1997), sensitive to the repetition of certain phenomena during the learning phase. Moreover, the sequential approach typical of deep-learning architectures, is also used by statistical analyses sensitive to the contextualization of terms. See, for instance, the analysis of repeated segments (Salem, 1987) or motifs(Longrée and Mellet, 2013).

Combining purely statistical approaches with artificial neural networks, this contribution allows us to identify key passages (either of a text or of an author) and linguistic features bringing us toward a deeper understanding of texts. it introduces a novel approach to explore the hidden layers of a convolutional neural network, trying to explain which are the relevant linguistic features used by the network to perform the classification task While the features which allow the standard statistical text-analysis to detect key passages are well known (the word specificity), the areas

of activation in deep learning seem to uncover new linguistic units that the traditional corpus semantics struggled to show.

The statistical text-analysis can be qualified as paradigmatic: the task is to tally the discrete elements, identified as being representative, over a given text portion (a sentence, a paragraph, en extract of *n* words). Some filters, either linguistic (for instance, only taking nouns into account) or statistical (for instance, only considering words with a positive z-score) can be applied to the discrete elements, in order to optimize the analysis. Note that the pertinence of these filters changes with the different corpora (different languages, different genres, specific speakers).

As for the deep-learning approach which may be qualified as syntagmatic: through the use of sliding windows, the goal is to uncover the areas in the texts that, in a particular context, allowed the neural network to distinguish one author from another or one text from another in a given corpus.

In social sciences, these areas of activation seem to fulfill the essential hermeneutics of the corpus semantics carried out by the machine (Rastier, 2007, 2011) Based on a the material substance of a text – since the machine needs tokens – for the linguist, the areas of activation are the key to a deeper interpretation of texts.

4.2 Perspectives

Given that areas of activation detected by deep-neural networks bring a dramatic improvement to our interpretation capabilities, this contribution needs to be extended in two directions. First, from a linguistic perspective, lemmatization and part-of-speech tagging should be implemented. Doing that would allow for a straight-forward description of the text, not only accounting for a lexical level, but also for a grammatical and syntactical level.

From a methodological perspective, in order to extract not only discriminative spatial linguistic markers (using CNNs), recurrent layers (RNN) might be introduced in the neural network. Those layers takes into account the temporal dimension of texts and produce a dynamic view. Future works might focus on the combination of both approaches to extract spatial linguistic patterns with CNNs and be able to analyze their sequential organization with RNNs.

References

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ducoffe, M., Precioso, F., Arthur, A., Mayaffre, D., Lavigne, F., and Vanni, L. (2016). Machine learning under the light of phraseology expertise: use case of

presidential speeches, de gaulle - hollande (1958-2016). In *Actes de JADT 2016*, pages 155–168.

Joulin, A., Grave, E., and Mikolov, P. B. T. (2017). Bag of tricks for efficient text classification. *EACL 2017*, page 427.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lafon, P. (1984). Dépouillements et statistiques en lexicométrie. In *Genève-Paris, Slatkine-Champion*.

Lebart, L. (1997). Réseaux de neurones et analyse des correspondances. In *Modulad, (INRIA Paris), 18*, pages 21–37.

Longrée, D. and Mellet, S. (2013). Le motif : une unité phraséologique englobante ? étendre le champ de la phraséologie de la langue au discours. *Langages 189*, pages 65–79.

Longrée, D., Mellet, S., and Lavigne, F. (2019). Construction cognitive d'un motif : cooccurrences textuelles et associations mémorielles. In *CogniTextes*, page http://journals.openedition.org/cognitextes/1202.

Mellet, S. and Longrée, D. (2009). Syntactical motifs and textual structures. In *Belgian Journal of Linguistics 23*, pages 161–173.

Mellet, S. and Longrée, D. (2012). Légitimité d'une unité textométrique : le motif. In *Actes de JADT 2012*, pages 715–728.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Quiniou, S., Cellier, P., Charnois, T., and Legallois, D. (2012). Fouille de données pour la stylistique : cas des motifs séquentiels émergents. In *Actes de JADT 2012*.

Rastier, F. (2007). Passages. *Corpus 6*, pages 25–54.

Rastier, F. (2011). *La mesure et le grain: sémantique de corpus*. Champion; diff. Slatkine.

Salem, A. (1987). Pratique des segments répétés. essai de statistique textuelle. *Paris : Klincksieck*.

Vanni, L., Ducoffe, M., Precioso, F., Mayaffre, D., Longree, D., and al. (2018). Text deconvolution saliency (tds) : a deep tool box for linguistic analysis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 548–557, Melbourne, Australia. Association for Computational Linguistics.

Vanni, L. and Mittmann, A. (2016). Cooccurrences spécifiques et représentations graphiques, le nouveau thème d'hyperbase. In *Actes de JADT 2016*, pages 295–305.

Annexe B

À propos

Quoi de mieux pour apprécier le contenu (linguistique) de cette thèse que de la confronter à la méthode qu'elle propose. Pour y parvenir, nous avons choisi d'analyser le texte de ce manuscrit en le comparant au corpus des JADTs[1] *composé de l'ensemble des actes en français parus entre 2000 et 2020, soit 20 ans et 11 éditions pour 368 (premiers) auteurs et un peu plus de 3,3 millions de mots.*

B.1. Contexte

La première analyse que nous proposons correspond au calcul de la cooccurrence généralisée. La fonction que nous appelons *correlats* dans Hyperbase présente (via une AFC) les associations spécifiques (*z-score*) qui existent entre les 300 substantifs les plus fréquents dans le texte. Cette analyse fait apparaitre les principaux thèmes abordés tout au long des chapitres. Plus particulièrement trois clusters se dessinent (figure B.1) :

Le premier fait référence à l'Analyse de Donnée Textuelle (ADT) statistique traditionnelle. Il y a d'une part des notions de linguistique de corpus tels que les *passages-clés*, *l'intertextualité* ou plus généralement le *corpus* associé au *lexique* et à la syntaxe (*pronom*, *verbe*, ...). D'autre part, des indices statistiques connus comme le *z-score* ou encore les méthodes qui reposent sur la *distribution* des mots et les *spécificité*s complètent se pôle ADT clairement identifié.

Le deuxième regroupement que nous pouvons faire concerne la méthode proposée par cette thèse. Le *deep learning* introduit ici un vocabulaire technique qui fait référence au fonctionnement des réseaux de neurones profonds, à savoir, des calculs sur *matrice*s, des représentations à base de *vecteur*s, ou encore des notions associées à l'*architecture* des modèles associés (*embedding*, *(dé)convolution*, *couche*s, *entrée*s, *sortie*s, ...). A noter la position sur le graphique du TDS, la mesure principale introduite dans cette thèse qui tend ici à se rapprocher de l'ADT. C'est en fait une volonté non dissimulée d'introduire solidement cette méthode aux côtés de celles qui ont déjà fait leurs preuves pour proposer une approche véritablement complémentaire.

1. « Les Journées internationales d'Analyse statistique des Données Textuelles (JADT) réunissent, tous les deux ans depuis 1990, 150 à 200 chercheurs travaillant dans les différents domaines concernés par les traitements automatiques et statistiques de données textuelles. Elles permettent aux participant-e-s de présenter leurs résultats, de confronter leurs outils, d'échanger sur leurs expériences pratiques et méthodologiques. » (voir `http://jadt.org`)

Enfin le dernier thème (la partie haute de la figure) souligne la singularité des développements informatiques associés à cette thèse. Principalement portée par le chapitre 8, l'implémentation informatique des méthodes proposées prend une place à part dans la liste des contributions proposées. De toute évidence, la fonction d'ingénieur de recherche occupée en parallèle de ces travaux est ici représentée et s'exprime par la production de *logiciel*s, *service*s et autres plateformes *web* qui proposent des *fonctionnalités* nouvelles aux *utilisateur*s.

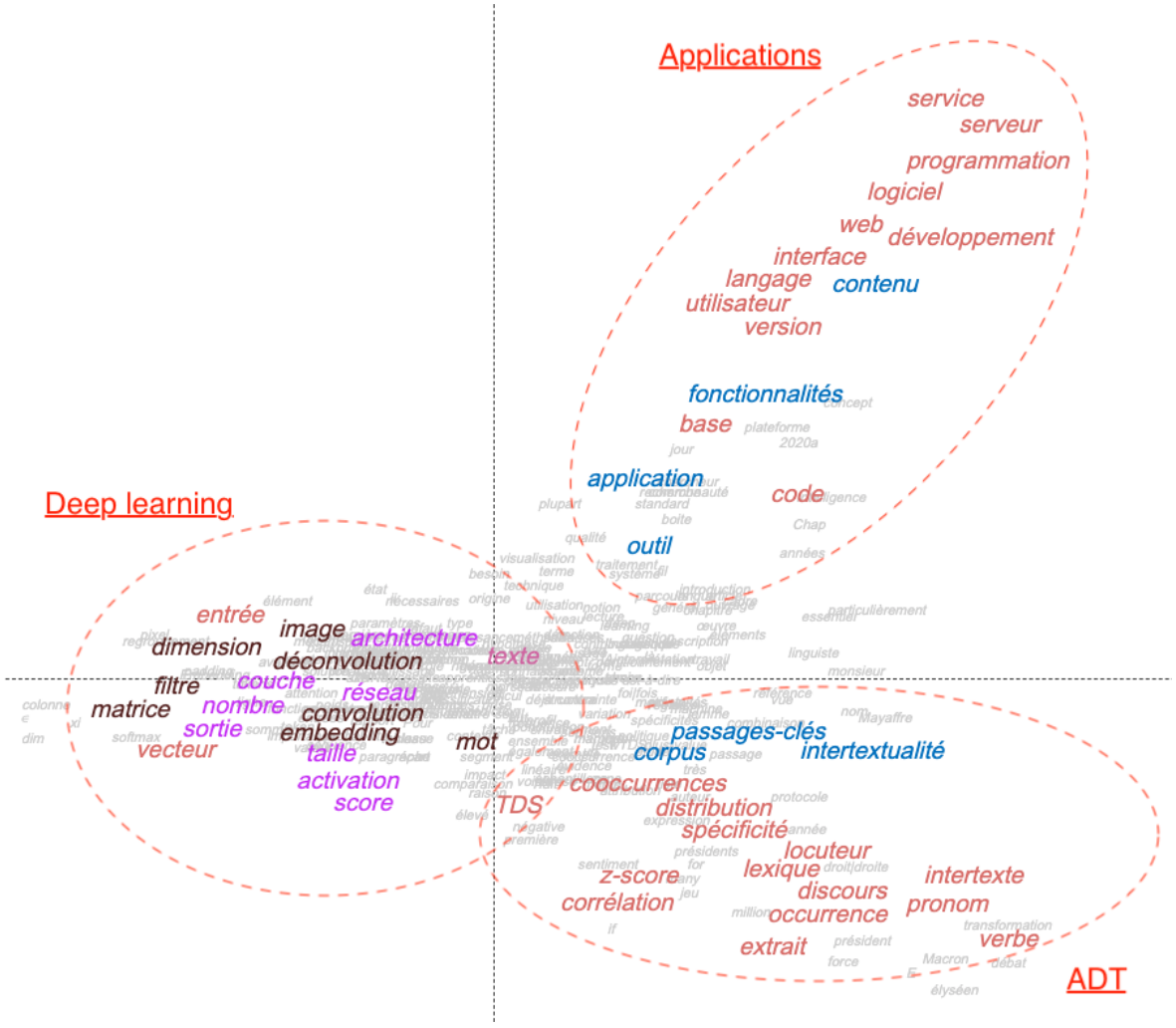


FIGURE B.1 – Analyse des cooccurrences des 300 substantifs les plus utilisés dans ce manuscrit de thèse.

B.2. Motifs profonds

La seconde analyse met en oeuvre toute la méthodologie proposée dans cette thèse. L'analyse des *motifs profonds* sur le manuscrit révèle des particularités lexicales, syntaxiques et syntagmatiques qui permet de l'identifier linguistiquement.

Le principe est le même que celui présenté dans les chapitres 5 et 6. Un réseau convolutionnel a été entrainé à classer cette thèse parmis les 572 articles contenu dans le corpus d'entrainement. L'analyse des *passages-clés* permet ensuite d'extraire les segments les plus représentatifs du texte et l'architecture *multi-channels* montre en utilisant le wTDS des *motifs* particuliers qui en marquent l'identité.

Un des extraits les plus caractéristiques pour le modèle (le deuxième par ordre hiérarchique) rassemble à lui seul un ensemble de motifs très significatifs :

> *[...] plus-value ADJ de la convolution sembler ADV attestée . En étant ADJ PRP interpréter dans un segment chaque mot dans son contexte , le deep learning associe les calcul ADJ connus (z-score , cooccurrence , segment ADJ ...) avec une NOM ADJ ou ADJ du texte pour en [...]*

(Extrait du chapitre 4, section 4.9)

Le discours est marqué visiblement par l'emploi de l'adjectif qui confère au texte une consonance technique par son aspect figé, comme dans l'expression *une approche séquentielle ou segmentale* (codé ici par l'enchainement $NOM + ADJ + ou + ADJ$). En évitant l'utilisation d'un article (défini ou indéfini) pour introduire le complément de nom, le discours se veut concis et formel. On note aussi l'enchainement *ADJ PRP* où la préposition qui suit l'adjectif ajoute une incidence qui lie l'adjectif au nom (ici le *deep learning*). Ce motif est d'ailleurs un des plus significatifs statistiquement (*z-score* +16.42 - figure B.2).

Parmi les autres marqueurs visibles, il y a des choix lexicaux plus évidents comme le mot *convolution* qui, même s'il a été déjà utilisé plusieurs fois aux JADTs depuis 2006, il reste fortement associé aux travaux proposés. Dans ce contexte technique, le syntagme *plus-value* et quant à lui une promesse de nouveauté de la méthode proposée. Cette promesse est néanmoins nuancée par l'utilisation du verbe *sembler*, une autre spécificité du texte (*z-score* +4.72) qui affirme le caractère exploratoire de la méthode (figure B.2).

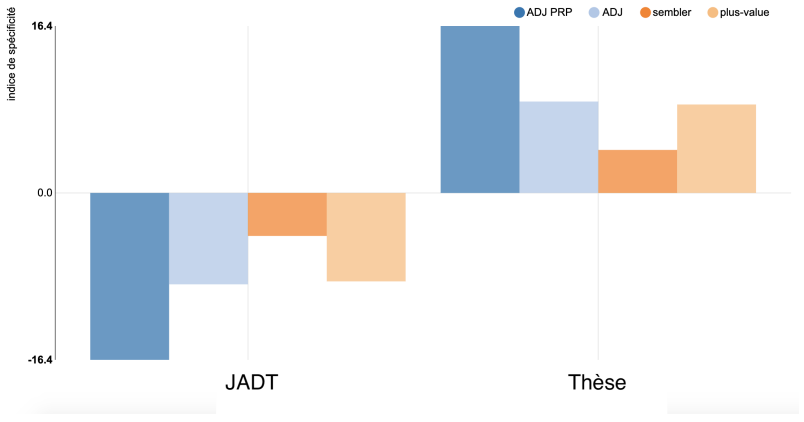


FIGURE B.2 – Spécificités lexicales et syntaxiques de ce manuscrit de thèse

Pour finir, l'extraction de l'ensemble des *motifs profonds* suggère une analyse linguistique qui nécessiterait de s'attarder plus longuement sur l'interprétation. Nous pouvons néanmoins suggérer quelques exemples qui illustrent encore une fois la capacité de la méthode à repérer des nouveaux observables complexes :

motifs	references
jusque ADV ADV	3.1, 3.2.5 ...
DET :ART NOM du [...] NOM ADJ	2.1, 5.3.2, 6.4 ...
PRO :DEM NOM [...] DET :ART NOM	1.1, 2.4.1 ...

TABLE B.1 – Exemples de motifs profonds extraits de ce manuscrit de thèse. Les crochets signifient que l'expression est composée de tokens non contigus (séparés au plus par une fenêtre de convolution)

Bibliographie

Adam, L. (2011). « La linguistique textuelle. Introduction à l'analyse textuelle des discours ». In : *A. Colin, coll. « Cursus »*.

Adel, Heike et Hinrich Schütze (2017). « Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification ». In : *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 1723–1729.

Baker, P. (2010). *Sociolinguistics and Corpus Linguistics*. Edinburgh sociolinguistics. Edinburgh University Press. isbn : 9780748627356. url : https://books.google.fr/books?id=8v7TOKyNGK4C.

Beaudouin, V. (2016). « Retour aux origines de la statistique textuelle : Benzécri et l'école française d'analyse des données ». In : *13es Journées internationales d'Analyse statistique des Données Textuelles (JADT2016)*.

Bojanowski, Piotr, Edouard Grave, Armand Joulin et Tomas Mikolov (2017). « Enriching word vectors with subword information ». In : *Transactions of the Association for Computational Linguistics* 5, p. 135–146.

Bouzereau, C. (2020). *Doxa et contredoxa dans la construction du territoire discursif du front national (2000-2015*.

Bres, J., P.-P. Haillet, S. Mellet, H. Nølke et L. Laurence Rosier/indexLaurence Rosier (2005). « Dialogisme, polyphonie : approches linguistiques ». In : *De Boeck*.

Brunet, E. (1978). *Le vocabulaire de jean giraudoux. Structures et evolution.* Genève : Slatkine.

— (1999). « Qui lemmatise dilemme attise ». In : *11e Rencontres linguistiques en pays rhénan*, p. 7–32.

Brunet, E., L. Lebart et L. Vanni (2021). « l'intelligence artificielle des textes ». In : Paris : Honoré Champion. Chap. Littérature et intelligence artificielle.

Brunet, E. et L. Vanni (2019). « Deep learning et authentification des textes ». In : t. XXIV. 1. Textes et Cultures, Institut Ferdinand de Saussure, p. 1–34.

Chattopadhay, Aditya, Anirban Sarkar, Prantik Howlader et Vineeth N Balasubramanian (2018). « Grad-cam++ : Generalized gradient-based visual explanations for deep convolutional networks ». In : *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, p. 839–847.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk et Yoshua Bengio (2014). « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation ». In : *EMNLP*.

Collobert, R. et J. Weston (2008). « A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning ». In : *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland : ACM, p. 160–167. isbn : 978-1-60558-205-4. doi : 10.1145/1390156.1390177. url : http://doi.acm.org/10.1145/1390156.1390177.

Collobert, R, J Weston, L Bottou, M Karlen, K Kavukcuoglu et P Kuksa (2011). « Natural Language Processing (Almost) from Scratch ». In : *Computing Research Repository - CORR* 12.

Dauphin, Yann N, Angela Fan, Michael Auli et David Grangier (2017). « Language Modeling with Gated Convolutional Networks ». In : *International Conference on Machine Learning*, p. 933–941.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee et Kristina Toutanova (2019). « 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)*, p. 4171–4186.

Ducoffe, M., D. Mayaffre, F. Precioso, F. Lavigne, L. Vanni et A. Tre-Hardy (2016). « Machine Learning under the light of Phraseology expertise : use case of presidential speeches, De Gaulle -Hollande (1958-2016) ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles*.

Feldman, R., and Sanger, J. (2007). *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data*. New York : Cambridge University Press.

Garreau, Damien et Ulrike von Luxburg (2020). « Explaining the explainer : A first theoretical analysis of LIME ». In : *arXiv preprint arXiv :2001.03447*.

Gisle, Andersen (2020). « Phraseology in a cross-linguistic perspective : A diachronic and corpus-based account ». In : *Corpus Linguistics and Linguistic Theory*, p. 1–25. doi : 10.1515/cllt-2019-0057.

Guaresi, M. (2020). « Décrire les textes politiques par le deep learning : à la recherche de nouveaux observables ». In : *15es Journées internationales d'Analyse statistique des Données Textuelles (JADT2020)*.

Guaresi, M., D. Mayaffre et L. Vanni (2022 (sous presse)). « Entre rupture et continuité, le discours du PCF (1920-2020 ». In : *Histoire et Mesure*.

Halliday, M. A. K. et R. Hasan (1976). *Cohesion in English*. London : Longman.

Hochreiter, Sepp et Jürgen Schmidhuber (1997). « Long short-term memory ». In : *Neural computation* 9.8, p. 1735–1780.

Ji, Yangfeng et Jacob Eisenstein (2014). « Representation learning for text-level discourse parsing ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. T. 1, p. 13–24.

Joulin, Armand, Edouard Grave et Piotr Bojanowski Tomas Mikolov (2017). « Bag of Tricks for Efficient Text Classification ». In : *EACL 2017*, p. 427.

Kalchbrenner, N, E Grefenstette et P Blunsom (2014). « A Convolutional Neural Network for Modelling Sentences ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. T. 1, p. 655–665.

Kalchbrenner, Nal, Edward Grefenstette et Phil Blunsom (2014). « A Convolutional Neural Network for Modelling Sentences ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. T. 1, p. 655–665.

Karpathy, Andrej, Justin Johnson et Li Fei-Fei (2015). « Visualizing and understanding recurrent networks ». In : *arXiv preprint arXiv :1506.02078*.

Kim, Y. (2014a). « Convolutional Neural Networks for Sentence Classification ». In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1746–1751.

— (2014b). « Convolutional Neural Networks for Sentence Classification ». In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1746–1751.

Labbé, C. et D. Labbé (2003). « La distance intertextuelle ». In : *Corpus*, p. 95–118.

Lafon, P. (1980). « Sur la variabilité de la fréquence des formes dans un corpus ». In : *Mots* 1, p. 127–165.

Lavigne, F et S Longrée D ans Mellet (2018). « Construction cognitive d'un motif : cooccurrences textuelles et associations mémorielles ». In : *CogniTextes*. Sous la dir. de CogniTextes. T. 18.

Lebart, L. and Salem, A. and Berry, L. (1998). *Exploring Textual Data*. Ed. Springer.

Lebart, L. (1997). « Réseaux de neurones et analyse des correspondances ». In : *Modulad (INRIA Paris)* 18, p. 21–37.

Lebart, L., B. Pincemin et C. Poudat (2019). *Analyse des données textuelles*. Presses de l'Université du Québec.

LeCun, Y., L. Bottou, Y. Bengio et P. Haffner (1998). « Gradient-Based Learning Applied to Document Recognition ». In : *IEEE*, p. 2278–2324.

Li, Jiwei, Xinlei Chen, Eduard Hovy et Dan Jurafsky (2015). « Visualizing and understanding neural models in NLP ». In : *arXiv preprint arXiv :1506.01066*.

Longrée, D. et S. Mellet (2013). « Le motif : une unité phraséologique englobante ? Étendre le champ de la phraséologie de la langue au discours ». In : *Langages* 189, p. 65–79.

Manning, Christopher D et Hinrich Schütze (1999). *Foundations of statistical natural language processing*. MIT press.

MARDAOUI, Dina et Damien GARREAU (2020). « An Analysis of LIME for Text Data ». In : *arXiv preprint arXiv :2010.12487*.

MAYAFFRE, D. (2002). « Les corpus réflexifs : entre architextualité et hypertextualité ». In : *Corpus*.

— (2012a). « Le discours présidentiel sous la Vème République. Chirac, Mitterrand, Giscard, Pompidou, de Gaulle ». In : *Presses de Sciences Po*.

— (2012b). *Mesure et démesure du discours. Nicolas Sarkozy (2007-2012)*. Presses de Sciences Po.

— (2021). *Macron ou le mystère du verbe*. Paris : L'Aube.

MAYAFFRE, D., C. BOUZEREAU, M. DUCOFFE, M. GUARESI, F. PRECIOSO et L. VANNI (2017). « Le vote disruptif. Les élections présidentielle et législatives de 2017 ». In : Paris : Presses SciencesPo. Chap. Les mots des candidats, de « allons » à « vertu », p. 129–152.

MAYAFFRE, D., M GUARESI et L. VANNI (2020). « Ces mots que Macron emprunte à Sarkozy. Discours et intelligence artificielle ». In : *Corpus* 21.

MAYAFFRE, D. et L. VANNI (2020). « Objectiver l'intertexte ? Emmanuel Macron, deep learning et statistique textuelle ». In : *15es Journées internationales d'Analyse statistique des Données Textuelles (JADT2020)*.

— (2021a). « Du texte profond. Textualité et deep learning ».

— (2021b). *L'intelligence artificielle des textes. Des algorithmes à l'interprétation*. Paris : Honoré Champion.

MELLET, S. et D. LONGRÉE (2009). « Syntactical motifs and textual structures ». In : *Belgian Journal of Linguistics*. T. 23, p. 161–173.

MIKOLOV, T., I. SUTSKEVER, K. CHEN, G. S. CORRADO et J. DEAN (2013). « Distributed representations of words and phrases and their compositionality ». In : *Advances in neural information processing systems*, p. 3111–3119.

MORETTI, F. (2013). *Distant Reading*. London/New-York : Verso, p. 244.

NOH, Hyeonwoo, Seunghoon HONG et Bohyung HAN (2015). « Learning deconvolution network for semantic segmentation ». In : *Proceedings of the IEEE International Conference on Computer Vision*, p. 1520–1528.

PENNINGTON, J., R. SOCHER et C. MANNING (2014). « Glove : Global vectors for word representation ». In : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, p. 1532–1543.

PRECHELT, L. (1998). « Early stopping-but when ? » In : *Neural Networks : Tricks of the trade*. Springer, p. 55–69.

RASTIER, F. (2004). « Enjeux épistémologiques de la linguistique de corpus. » In : *Texto !*

— (2007). « Passages ». In : *texto !* 13.

— (2020). « Sémiosis et métamorphoses ». In : *Semiotica*, p. 145–162.

— (2021). « l'intelligence artificielle des textes ». In : Paris : Champion. Chap. DATA VS CORPORA, p. 201–244.

RIBEIRO, M. Tulio, S. SINGH et C. GUESTRIN (2016). « Why should i trust you ? : Explaining the predictions of any classifier ». In : *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, p. 1135–1144.

ROSENBLATT, F. (1958). « The perceptron : A probabilistic model for information storage and organization in the brain. » In : *Psychological Review* 65.6, p. 386–408.

RUDER, S, P GHAFFARI et J BRESLIN (2016). « Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution ». In :

RUGGIA, S. (2019). « Le deep learning : un outil pour la didactique du FLE ? » In : *Dialettica pedagogica* 1, p. 79–106.

RUGGIA, S. et L. VANNI (2021). « DeepFLE : la plateforme pour évaluer le niveau d un texte selon le CECRL ». In : *3 e Congrès Européen de la FIPF*. Athènes.

SALEM, A. (1991). « Les séries textuelles chronologiques ». In : *Histoire et Mesure*.

SELVARAJU, Ramprasaath R, Michael COGSWELL, Abhishek DAS, Ramakrishna VEDANTAM, Devi PARIKH et Dhruv BATRA (2017). « Grad-cam : Visual explanations from deep networks via gradient-

based localization ». In : *Proceedings of the IEEE international conference on computer vision*, p. 618–626.

Vanni, L., M. Corneli, D. Longrée, D. Mayaffre et F Precioso (2020a). « Hyperdeep : deep learning descriptif pour l'analyse de données textuelles ». In : *15es Journées internationales d'Analyse statistique des Données Textuelles (JADT)*.

— (2020b). « Key passages : from statistics to deep learning ». In : *Text Analytics. Advances and Challenges*, p. 41–54.

Vanni, L., M. Corneli, F. Precioso et D. Mayaffre (2022). « Text Class Activation Map (T-CAM) ».

Vanni, L., M. Ducoffe, D. Mayaffre, F Precioso, D. Longrée, V. Elango, N. Santos, J. Gonzalez, L. Galdo et C. Aguilar (2018). « Text Deconvolution Saliency (TDS) : a deep tool box for linguistic analysis ». In : *56th Annual Meeting of the Association for Computational Linguistics (ACL)*. Melbourne, p. 548–557.

Vanni, L., X. Luong et D. Mayaffre (2014). « Arbre et co-occurrences Nouvel outil logométrique sur le net. Application au discours de François Hollande ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles* 1, p. 639–649.

Vanni, L., D. Mayaffre et D. Longrée (2018). « ADT et deep learning, regards croisés. Phrases-clefs, motifs et nouveaux observables ». In : *14es Journées internationales d'Analyse statistique des Données Textuelles*. Rome, p. 459–466.

Vanni, L. et A. Mittmann (2016). « Cooccurrences spécifiques et représentations graphiques, le nouveau "Thème" d'Hyperbase ». In : *12es Journées internationales d'Analyse statistique des Données Textuelles*. T. 1, p. 295–305.

Vanni, L. et F Precioso (2021). « L'intelligence artificielle des textes ». In : Paris : Honoré Champion. Chap. Deep learning et description des textes Architecture méthodologique.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser et Illia Polosukhin (2017). « Attention is All you Need ». In : *NIPS*.

Viprey, Jean-Marie (2006). « Structure non-séquentielle des textes ». In : *Langages* 163, p. 71–85.

Wen, Tsung-Hsien, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes et Steve Young (2017). « A Network-based End-to-End Trainable Task-oriented Dialogue System ». In : *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*. T. 1, p. 438–449.

Wiegand, M., J. Ruppenhofer et T. Kleinbauer (2019). « Detection of Abusive Language : the Problem of Biased Datasets. » In : *The North American Chapter of the Association for Computational Linguistics*. Minneapolis.

Yin, Wenpeng, Katharina Kann, Mo Yu et Hinrich Schütze (2017). « Comparative study of cnn and rnn for natural language processing ». In : *arXiv preprint arXiv :1702.01923*.

Yuan, X., P. He, Q. Zhu et X. Li (2019). « Adversarial Examples : Attacks and Defenses for Deep Learning ». In : *IEEE Transactions on Neural Networks and Learning Systems*. T. 30, p. 2805–2824.

Zeiler, Matthew D et Rob Fergus (2014). « Visualizing and understanding convolutional networks ». In : *European conference on computer vision*. Springer, p. 818–833.

Zhou, Bolei, Aditya Khosla, Agata Lapedriza, Aude Oliva et Antonio Torralba (2016). « Learning deep features for discriminative localization ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2921–2929.

Index