Rapport de Stage de Master

Machine Learning & Word Embeddings pour la classification et l'analyse d'articles encylopédiques

Khaled CHABANE

Université Lumière Lyon 2,
ICOM

Formation
Master en Informatique - Data Mining

Année Universitaire 2020-2021

Encadré par : Mr. Ludovic MONCLA
Mr. Julien Velcin

Remerciements

Mes remerciements à Monsieur Ludovic MONCLA pour son encadrement et ses conseils tout au long de ce projet.

Mes remerciements à Madame Alice BRENON pour son assistance et son aide.

Mes remerciements au laboratoire LIRIS pour m'avoir accueilli au sein de l'équipe DM2L, que je remercie également pour l'accueil et les moyens qu'on m'a fourni pour le travail.

Mes remerciements à Monsieur Julien VELCIN pour avoir répondu présent à mes emails, son cours m'a été important pour le déroulement de mon stage.

Résumé

Ce stage s'inscrit dans le cadre du projet GEODE[1] financé par le LabEx ASLAN[2] et dont l'objectif est d'étudier dans un corpus de quatre encyclopédies françaises les changements survenus dans les discours géographiques entre 1750 et nos jours. Pour cela, les méthodes de classification semi-supervisée des textes, de génération de modèles de langues et de repérage automatique de routines discursives seront développée. EN particulier, lors de ce stage nous nous sommes intéressé à l'Encyclopédie de Diderot et d'Alembert (1751-1772). Cette encyclopédie représente une ressource riche de connaissances issus du siècle des Lumières. Cette collection de contributions comprend une organisation synthétique et cohérente où chaque article est affecté à une classe de connaissance (domaine). Cette classification joue un rôle important notamment pour l'accès à l'information, l'exploration et l'analyse de ces connaissances.

Le présent travail a pour but de développer un classifieur, en utilisant l'apprentissage automatique et des techniques de plongement de mots, capable d'identifier la classe des articles de l'encyclopédie de manière efficace. Le jeu de données composés des articles de l'Encyclopédie déjà classés nous permet d'entrainer des modèles par apprentissage supervisé. L'objectif sera de pouvoir exploiter ces modèles pour classer les articles non classés (environ 20 000) mais également de classer les articles d'autres encyclopédies qui ne possèdent pas d'information de classification. Dans ce travail, après une étape d'extraction, de préparation et de nettoyage des données nous avons pu expérimenter et comparer différentes approches de classification. Notre méthodologie s'est intéressée à comparer les différentes approches de vectorisation (sac de mots, tf-idf, plongement de mots) et de classification (Naive Bayes, Logistic Regression, Random Forest, SGD, SVM). Nous avons également pu expérimenter les modèles de langues pré-entrainé (BERT, CamemBERT). Les résultats sont encourageants et montre une grande disparité entre les classes. L'ensemble du code développé dans le cadre de ce stage est disponible en open-source sur le repository Gitlab du projet GEODE[3].

Mots clé : Classification de texte, apprentissage automatique, extraction de caractéristiques, plongement de mots

[1] https://geode-project.github.io/

[2] https://aslan.universite-lyon.fr/

[3] https://gitlab.liris.cnrs.fr/geode/EDdA-Classification

Table des matières

Liste des figures

Liste des tableaux

1 Introduction

L'objectif de ce stage est de développer des modèles de classification des articles pour différentes encyclopédies (l'Encyclopédie de Diderot et d'Alembert (1751-1772), La Grande Encyclopédie, l'Encyclopædia Universalis et Wikipedia). Ce travail s'intéressera en particulier à la génération de modèles pour la classification des articles par l'expérimentation et la génération de modèles de langue permettant une représentation informatique des articles afin de pouvoir réaliser une analyse et une comparaison des différents corpus. L'exploitation des méthodes d'apprentissage supervisé ou d'apprentissage profond est un point important pour la réalisation de ce projet.

Le jeu de données (corpus numérique) de l'Encyclopédie de Diderot et d'Alembert qui sera notre principal corpus d'étude est proposé par deux organismes différents : ARTFL[4] (*American and French Research on the Treasury of the French Language*) et ENCCRE[5] (Édition Numérique Collaborative et CRitique de l'Encyclopédie). Dans le cadre d'une collaboration, les membres du projet GEODE ont accès aux données fournis par l'ARTFL. La base de données proposée contient 21.7 million de mots, 254,000 formes uniques, 18,000 pages de texte, 17 volumes d'articles, 77 085 articles, et 11 volumes de gravures. Dans notre travail, nous nous intéressons uniquement aux 17 volumes de textes.

Le système de classification des articles diffère d'une source à l'autre. Par exemple, ENCCRE parle de désignants. Il s'agit des indications figurant dans le texte, après la vedette (titre de l'article) et généralement entre parenthèses. La figure 1 montre l'article *Abyde*[6] issus de la plateforme Philologic de l'ARTFL. On voit bien dans cet exemple la présence du désignant (Géog. anc.) pour indiquer qu'il s'agit d'un article de géographie ancienne.

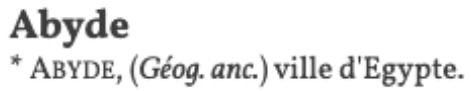


Figure 1: Exemple d'article avec la présence d'un désignant

Il existe une multitude de formulations très variées et non normalisées de ces désignants, plus de 7 000 occurrences. Afin de réduire ce nombre ENCCRE a procédé à plusieurs opérations pour produire des désignants explicités telles que la suppression des variations typographiques, la suppression des formules d'introduction, la suppression des abbréviations, la modernisation de l'orthographe et l'uniformisation des formules proches. À la suite des ces opérations, nous avons 2 160 désignants explicités. Ce nombre reste trop élevé. Une autre étape a alors consisté à construire une liste de domaines. Ils obtiennent ainsi 327 domaines qu'ils ont par la suite regroupés en 44 ensemble de domaines. La difficulté au-delà de normaliser et réduire le nombre de classes est que tous les articles ne contiennent pas de désignants. Ainsi, 12 635 articles ne contiennent pas de désignants et ce nombre est réduit à 2 392 après une étape de correction manuelle qui a consisté à assigner des désignants implicites récupérables par rapport au contexte et au contenu des articles. Du côté de l'ARTFL, le traitement réalisé a été très simple et sans (ou avec très peu) d'intervention humaine. Sur les 77 085 articles proposés par l'ARTFL, 55 248 sont associés à une classe et on retrouve 2 620 classes (normalisées). Il y a ainsi environ 20 000 articles non classés. Certains travaux menés par les équipes de l'ARTFL se sont déjà intéressés à la classification automatique des articles afin de "classer les non-classés" [Horton et al., 2009, Roe et al., 2016]. Notre objectif dans ce stage sera de mener de nouvelles

[4] https://artfl-project.uchicago.edu/

[5] http://enccre.academie-sciences.fr/encyclopedie/

[6] https://artflsrv03.uchicago.edu/philologic4/encyclopedie1117/navigate/1/350/

expérimentations afin d'aller plus loin, de tester des approches plus récentes et d'essayer d'améliorer les performances de classification.

Ce rapport de stage est structuré comme suit : la section 2 présente la structure d'accueil et le projet GEODE dans lequel s'inscrit ce stage. La section 3 fait un état de l'art des travaux sur la classification de textes. La section 4 décrit la méthodologie et le travail de conception et de développement réalisé. La section 5 présente les différentes expérimentations et les résultats obtenus. Enfin, la section 6 conclut ce rapport et présente les limites et perspectives.

2 Présentation de l'organisme de stage

2.1 Le LIRIS

Le laboratoire d'Informatique en Image et Systèmes d'Information (LIRIS) est une unité mixte de recherche en informatique ayant pour tutelles l'INSA Lyon, le CNRS, l'Université Claude Bernard Lyon 1, l'Université Lumière Lyon 2 et l'Ecole Centrale de Lyon. Il compte environ 120 membres permanents et 160 doctorants, et a pour principal champ scientifique l'Informatique et plus généralement les Sciences et Technologies de l'Information. Le LIRIS est né en 2003 à la suite du regroupement de 3 laboratoires de recherche lyonnais travaillant dans le domaine des systèmes d'information (LISI, LIGIM et RFV).

Le laboratoire possède 14 équipes de recherche, divisées dans 6 différents pôles, comme le montre le tableau 1.

Pôle	Équipes
Vision Intelligente et Reconnaissance Visuelles	Imagine
Géométrie et Modélisation	M2DisCo et GeoMod
Science des Données	BD, DM2L et GOAL
Interactions et Cognition	SMA,TWEAK et SICAL
Services, Systèmes Distribués et Sécurité	DRIM et SOC
Simulation, Virtualité et Sciences Computationnelles	Beagle, R3AM et SAARA

Table 1: Les équipes du LIRIS

Dans le cadre de ce stage, je travaille au sein de l'équipe DM2L sous l'encadrement de Monsieur Ludovic Moncla, maître de conférences en informatique à l'INSA Lyon et de Madame Alice Brenon doctorante en informatique. Les thématiques de recherche de mes tuteurs s'articulent autour des méthodes de traitement automatique du langage naturel et plus particulièrement de l'extraction d'informations géographiques à partir de textes par des techniques d'intelligence artificielle.

L'équipe DM2L concentre ses études dans des méthodes et outils pour la découverte d'information à partir de données, en faisant usage de techniques automatiques ou semi-automatiques. Ses principaux domaines de recherche sont le Data Mining et le Machine Learning. L'objectif principal de l'équipe est le développement de nouvelles méthodes et algorithmes de data mining et de machine learning, mais aussi à leur utilisation sur des données réelles pour extraire des connaissances et résoudre des problèmes concrets.

2.2 Le projet GEODE

Ce stage s'inscrit dans le cadre du projet GEODE[7] financé par le LabEx ASLAN. Il s'agit d'un projet pluridisciplinaire qui regroupe des chercheurs en informatique, linguistique, géographie et histoire travaillant au sein de différents établissements et laboratoires. Le projet GEODE s'intéresse à plusieurs aspects tels que la préparation des corpus (homogénéisation des formats, corrections, annotations) afin que le contenu de chaque encyclopédie puisse être traité automatiquement, la conception d'algorithmes adaptés pour l'analyse automatique et la recherche d'information géo-sémantiques. En particulier au développement de modèles linguistiques adaptés à l'analyse diachronique du discours géographique dans un corpus de documents textuels publiés entre le XVIIIe et le XXIe siècle. La méthodologie de l'équipe du projet est basée sur le développement d'une chaîne de traitement composée de différentes tâches qui nécessitent des ressources spécifiques (documents annotés, modèles linguistiques, ressources géographiques, etc.).

L'objectif principal du projet est d'étudier dans un corpus de quatre encyclopédies françaises les changements survenus dans les discours géographiques entre 1750 et nos jours. Pour cela, des méthodes de classification semi-supervisées des textes, de génération de modèles de langues et de repérage automatique de routines discursives sont développées et utilisées.

3 Travaux sur la classification de textes

La classification de texte ou la catégorisation de documents est un domaine d'études très émergent. Il existe plusieurs cas d'application de ce concept comme l'étiquetage des produits qui permet une meilleure navigation dans les sites e-commerce [Zhang and Paramita, 2019], les filtres de spams [Dada et al., 2019], ou encore la détection de la satisfaction des clients à travers leurs avis.

En utilisant des méthodes de traitement automatique du langage naturel, les classifieurs de texte peuvent analyser automatiquement le texte, puis attribuer un ensemble de catégories prédéfinies en fonction de son contenu. L'utilisation de l'apprentissage automatique a reçu beaucoup d'intérêt dans plusieurs domaines, y compris le traitement automatique du langage naturel. Beaucoup d'algorithmes d'apprentissage ont permis d'avoir de meilleurs résultats notamment dans la classification de texte.

Bien que nous disposons d'une panoplie de techniques très variés pour effectuer une classification de texte par apprentissage automatique, les étapes, quant à elles, restent bien définies comme le montre la figure 2.

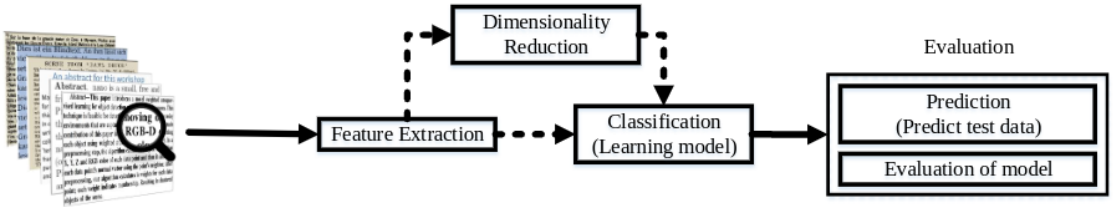


Figure 2: Les étapes pour la classification de texte [Kowsari et al., 2019])

On sait que les données textuelles sont des données non structurées et que les algorithmes utilisent le plus souvent des données numériques sous forme de vecteurs, d'où l'objectif de cette première étape dite "*Feature Extraction*". C'est une étape importante, où il s'agit d'extraire les caractéristiques des textes pour

[7] https://geode-project.github.io/

avoir une représentation vectorielle afin d'entraîner les classifieurs. Il existe deux familles de méthodes de représentation de documents. La première est la famille des mots pondérés. il y a la représentation par sac de mots "*Bag of Words* [Zhang et al., 2010], où chaque document est représenté par l'ensemble de ses mots qui appartiennent au vocabulaire et à chaque mot se voit attribuer sa fréquence dans le document, c'est à dire, que pour un document donné, sa représentation vectorielle sera les fréquences des termes du vocabulaire dans ce même document. Il existe aussi la représentation TF-IDF [Yun-tao et al., 2005] (Term frequency — Inverse document frequency), le concept TF désigne la fréquence du mot dans le document par rapport au nombre de mots de celui-ci, le concept IDF est une mesure d'unicité, l'idée derrière ce concept est qu'un mot qui apparaît beaucoup dans un corpus ne rajoute pas beaucoup d'informations dans un document.

Ce type de représentation capture des informations structurelles des documents car il prend en compte la mesure de fréquence et non la sémantique ou le contexte. Ce deuxième cas de figure, par contre, est capturé par une autre famille de représentation, les plongements de mots. Il existe beaucoup de techniques de plongement comme Word2Vec [Mikolov et al., 2013] et Glove [Pennington et al., 2014] qui sont fréquemment utilisées pour la représentation des mots sous forme de vecteurs et sont plus performants que les techniques précédentes dans plusieurs tâches comme la classification.

Dans l'étape de classification "*Learning model*", nous y trouvons beaucoup de méthodes comme le K-plus proches voisins, le machine à vecteurs de support, et le Naive Bayes [Sreemathy and Balamurugan, 2012, Colas and Brazdil, 2006, Xu, 2018], ainsi que d'autres méthodes d'apprentissage classiques comme la régression logistique, les arbres de décision ou les forêts aléatoires. Les modèles récents d'apprentissage profond ont apporté un progrès significatif dans la classification de textes, il existe des techniques considérables dans la réalisation d'une telle tâche, chacune a sa particularité. On y trouve énormément de travaux comme l'indique le tableau 2 qui avance les différentes familles de classifieurs à base de réseaux profonds avec des exemples de travaux récents de chaque famille. On rencontre des travaux qui se sont basés sur des réseaux perceptron multicouches qui sont des réseaux de neurones très simples dont les résultats sont moins attrayants que ceux d'autres modèles. Il existe aussi des travaux à base de réseaux récurrents (RNN), celles-ci sont destinés à capturer les dépendances de mots et les structures de textes. Une autre famille de réseaux dits convolutifs (CNN) qui sont destinés principalement aux traitements d'image ont pu donner de très bons résultats sur des tâches de traitement de texte. La raison derrière ces bonnes performances est que les CNN, par leur couches de convolution, cherchent à trouver des *patterns* de localité comme le cas des pixels adjacents dans des images. Le texte affiche également une localité et une information spatiale comme par exemple un adjectif est toujours suivi par un autre adjectif ou un nom, ce qui explique le bon rendement des CNN dans certaines tâches de NLP où la localité spatiale est importante comme la classification de texte.

D'autres modèles à base du mécanisme d'attention ont émergé et ont surpassé des modèles des familles cités antérieurement. La motivation derrière ces modèles est de trouver quel poids d'importance les mots peuvent avoir dans un document, ou plus précisément, ces modèles peuvent apprendre quelle est la contribution de chaque mot et de chaque phrase sur la classification.

Bien que les textes en langue naturelle présentent un ordre séquentiel, ils contiennent également une structure qui définit les relations syntaxiques et sémantiques entre les mots des phrases. Des réseaux comme les RNN ne sont pas capables de gérer les dépendances à longue distance. Pour résoudre ce problème une modélisation du texte sous forme de graphe et une définition particulièrement des liaisons des noeuds qui représentent les liens entre mots et documents ont été proposés. Pour bénéficier des avantages d'une telle structure et pouvoir faire du traitement dessus, certains travaux ont exploité les *graph neural networks* (GNN) et les résultats sont très concluants [Scarselli et al., 2008].

L'un des obstacles dont souffrent les réseaux décrits précédemment comme les RNN est le traitement séquentiel de long textes à cause de longues dépendances entre les mots, les CNN exigent un coût de calcul

énorme fortement corrélé avec la longueur du texte. Les Transformers [Vaswani et al., 2017] pallient ce problème et permettent également une parallélisation beaucoup plus poussée que d'autres modèles ainsi ils rendent possible l'entraînement de très gros modèles sur de grandes quantités de données.

Par ailleurs, Il y a d'autres techniques de classification de texte fondées sur des architectures et des moyens un peu plus poussés comme l'apprentissage par renforcement [Sutton and Barto, 2018], *Memory Networks* [Weston et al., 2014], ou les *Siamese neural networks* [Bromley et al., 1993]

Famille de réseaux	Travaux
Classification à base de réseaux MLP	DAN [Iyyer et al., 2015]
Classification à base de RNN	TopicRNN [Dieng et al., 2016]
Classification à base de CNN	VDCNN [Conneau et al., 2016]
Classification à base de mécanisme d'attention	BI-Attention [Zhou et al., 2016]
Classification à base de réseaux Transformers	ELMO [Peters et al., 2018], Bert, GPT
Classification à base de GNN	TextGCN [Yao et al., 2019], DGCNN [Peng et al., 2018]
Autres	Siamese LSTM [Shih et al., 2017],Memory Network [Zeng et al., 2018]

Table 2: Travaux de classification de texte basés sur l'apprentissage profond [Li et al., 2020])

Après cet aperçu de l'état de l'art sur la classification de texte, où les différentes étapes du processus sont mis en évidence avec la présentation des différentes techniques et travaux de chacune des étapes. Il est à présent question de choisir les méthodes pour répondre au besoin et aux contraintes des données.

4 Méthodologie et entraînement de modèles de classification

4.1 Présentation des données

Le jeu de données numérisé de l'Encyclopédie auquel nous avons accès est fournit par l'ARTFL. Les données sont organisées par tome (ou volume) puis par article (ou numéro). Il s'agit de fichiers XML-TEI, chaque article est contenu dans un fichier TEI et l'ensemble des fichiers d'un même volume sont regroupés dans un répertoire (voir figure 3). Le jeu de données se compose de 17 volumes et d'environ 70 000 articles.

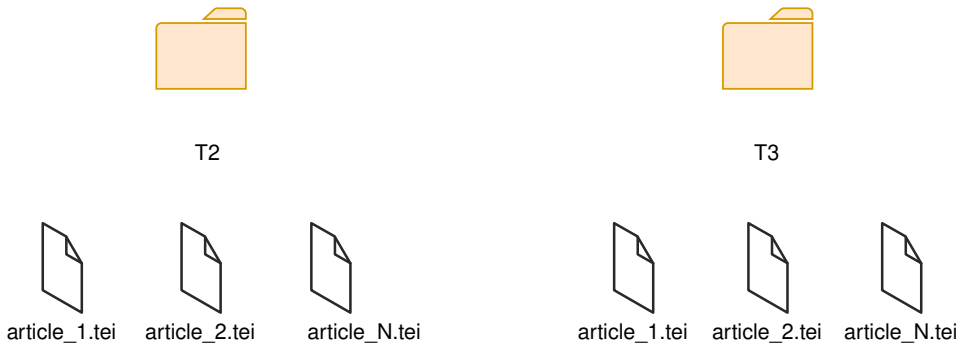


Figure 3: Organisation des données

Un fichier TEI va, dans ce cas, comprendre plusieurs balises pour exprimer les méta-données c'est à dire les propriétés de l'article et le contenu qui est le texte (voir figure 4).

```
−<TEI.2>
  +<teiHeader></teiHeader>
  −<text>
    −<body>
      −<div1 n="349" id="0:249:1" vol="1">
          <index type="head" value="Abyde"/>
          <index type="objecttype" value="arts"/>
          <index type="author" value="Diderot"/>
          <index type="shortauth" value="Diderot"/>
          <index type="kafnum" value="39"/>
          <index type="kafauth" value="Diderot, Denis (1713-1784)"/>
          <index type="attribution" value="signed"/>
          <index type="class" value="Geog. anc."/>
          <index type="normclass" value="Géographie ancienne"/>
          <index type="englishclass" value="Ancient geography"/>
          <index type="generatedclass" value="Géographie ancienne"/>
          <index type="pos" value="NA"/>
          <head>Abyde</head>
        −<p>
            <sc>* Abyde</sc>
            , (
            <i>Géog. anc.</i>
            ) ville d'Egypte.
          </p>
        </div1>
      </body>
    </text>
</TEI.2>
```

Figure 4: Exemple de fichier TEI

4.2 Description de la solution

Dans le but de réaliser la classification, on conçoit une méthodologie cohérente pour aboutir à un travail ordonné en étapes bien définies. La figure 5 met en évidence le processus et ses étapes qu'on expliquera une par une dans les points ci-après.

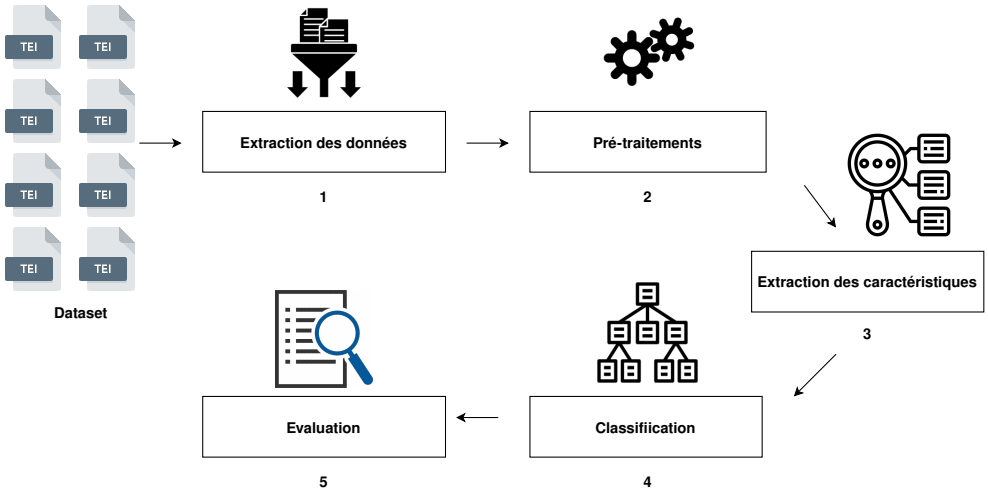


Figure 5: Processus de conception du classifieur

4.2.1 Extraction des données

La première étape du travail consiste à extraire les données de ces fichiers TEI afin de pouvoir les exploiter. Le but de cette étape est de construire un dataframe à partir des fichiers TEI. Pour mieux expliquer cela, chaque fichier TEI sera une ligne dans le dataframe, donc un objet python. L'utilisation d'un type de donnée comme le dataframe est pour la facilitation de la manipulation de ces données.

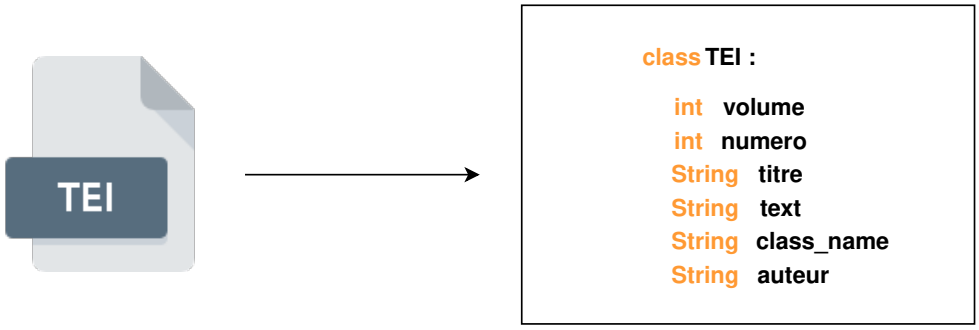


Figure 6: Passage d'un fichier TEI à un objet

Cette étape est impérative car elle permet de faire l'extraction du contenu des fichiers tei pour la classification ainsi que la sélection et l'extraction de méta-données importantes pour une utilisation ultérieure.

4.2.2 Pré-traitement

L'étape de pré-traitement est une étape importante avant la construction du modèle de classification. La qualité des résultats peuvent dépendre de la manière dont les données ont été traitées. Dans cette étape, plusieurs opérations ont été faites et sont listées comme suit :

- **Vérification et Suppression des valeurs nulles.**
- **Suppression des désignants**

  Il faut savoir que les textes extraits des fichiers TEI comprennent pour la plupart à leur début une abréviation ou le mot complet qui désigne la classe à laquelle l'article appartient. Ces désignants ne font pas à proprement partie du texte brut de l'article et peuvent donc rajouter un biais à la classification (en particulier lorsqu'il est manquant).

- **Rééquilibrage des classes**

  Comme évoqué en introduction, une difficulté est de savoir quelle classification utiliser comme label pour notre apprentissage. Afin d'avoir un nombre de classe limité nous avons choisi la classification faite par ENCCRE en 44 ensembles de domaines (voir table 3) et 327 domaines plutôt que les 2 620 classes normalisées répertoriée par l'ARTFL. Nous avons en réalité regroupé les 7 classes de métiers au sein d'une même classe et nous considérons donc 38 ensemble de domaines.

Agriculture - Economie rustique	Commerce	Médailles	Minéralogie
Anatomie	Droit - Jurisprudence	Médecine - Chirurgie	Monnaie
Antiquité	Economie domestique	Mesure	Musique
Architecture	Géographie	Métiers de l'alimentation	Pêche
Arts et métiers	Grammaire	Métiers du bois	Pharmacie
Beaux-arts	Histoire	Métiers du cuir et des peaux	Philosophie
Belles-lettres - Poésie	Histoire naturelle	Métiers du métal, du minéral et dérivés	Physique
Blason	Jeu	Métiers du papier	Politique
Caractères	Maréchalerie - Manège	Métiers du tissu et de l'habit	Religion
Chasse	Marine	Métiers et matières autres	Spectacle
Chimie	Mathématiques	Militaire (Art) - Guerre - Arme	Superstition

Table 3: Liste des 44 ensembles de domaines répertorié par ENCCRE

Cependant malgré un nombre de classes réduit, la figure 7 montrent que la répartition des instances dans les classes n'est pas équilibrée. La classe "Géographie" par exemple comprend 13 313 instances alors que la seconde classe du classement "Droit-Jurisprudence" comprend seulement 7 488, et certaines classes comprennent un nombre d'instance inférieur à 1 000 comme le cas de "Mathématiques", "Musique" et "Arts et métiers".

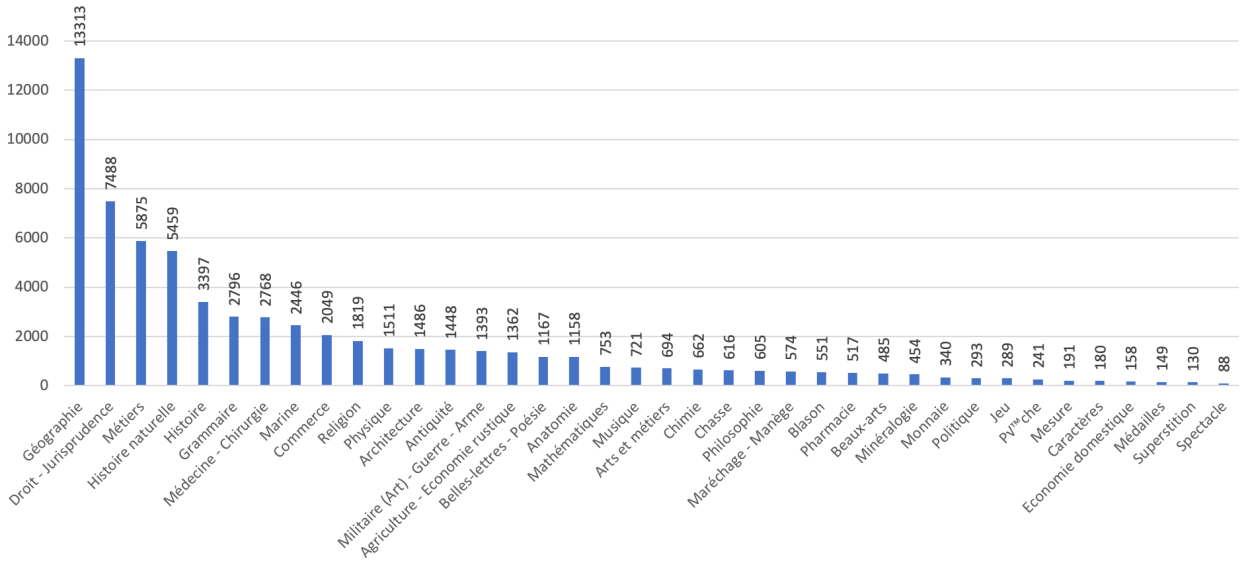


Figure 7: Distribution des instances de classes selon ENCCRE

Pour pallier à ce problème et dans le but d'obtenir une classification juste, il est nécessaire de procéder à un rééquilibrage des classes et ceci selon une étude empirique et en fixant un seuil sur le nombre maximum d'instance qu'une classe peut avoir.

- **Utilisation d'un tokeniseur approprié**

  Avant de procéder à l'extraction des caractéristiques, il est primordial de faire un *Data Cleaning*, c'est à dire faire une tokenisation propre à la langue du texte qui permet de récupérer un bon découpage des mots du texte, enlever les mots vides, dans certains cas enlever les mots trop fréquents ainsi que faire une lemmatisation ou une stemmatisation.

- **Standarisation des documents**

  Faire une suppression des articles dont le nombre de *tokens* ne dépasse pas un certains seuil (articles trop courts) qu'on fixera par la suite à 25. Après la tokenisation des documents, on pourra dès alors supprimer les tokens dont le nombre d'occurrences dépasse un certain seuil, ou au contraire n'atteint pas un seuil minimal.

4.2.3 Extraction des caractéristiques

L'étape d'extraction des caractéristiques est consacrée à la représentation des documents sous format initiales textuelles vers un format vectoriel qui permet de capturer le plus possible la sémantique et le contexte. Pour cela nous exploiterons plusieurs techniques de vectorisation, qu'il s'agisse des techniques de pondération de mots ou de plongement de mots.

Sac de mots La technique de vectorisation appellée "sac de mots" (*bag of words*) permet de capturer la particularité structurelle du document grâce à la notion de fréquence. On construit un vocabulaire à partir des documents tokenisés. Puis, pour chaque document, la technique construit une représentation vectorielle qui fait correspondre à chaque mot du vocabulaire sa fréquence dans le document adéquat comme le montre le tableau 4.

	Abyde	ville	Egypte
01-202	0	0	2
01-349	1	1	0
01-350	1	1	1

Table 4: Exemple de représentation avec la technique sac de mots

L'inconvénient d'une telle méthode est que si le vocabulaire est grand, les vecteurs des documents comprennent beaucoup de valeurs nulles et les vecteurs sont aussi grands que le vocabulaire.

TF-IDF ou Term Frequency–Inverse Document Frequency, est une statistique numérique pour refléter l'importance d'un mot dans un document. On obtient une représentation similaire que celle de "Sac de mots" comme dans le tableau 5 mais au lieu de prendre en compte la fréquence, on prend en compte le $(tf\_idf)_{t,d}$, à savoir le tf_idf d'un terme t dans un document d qui est exprimé par l'équation 1 suivante, où # désigne "nombre":

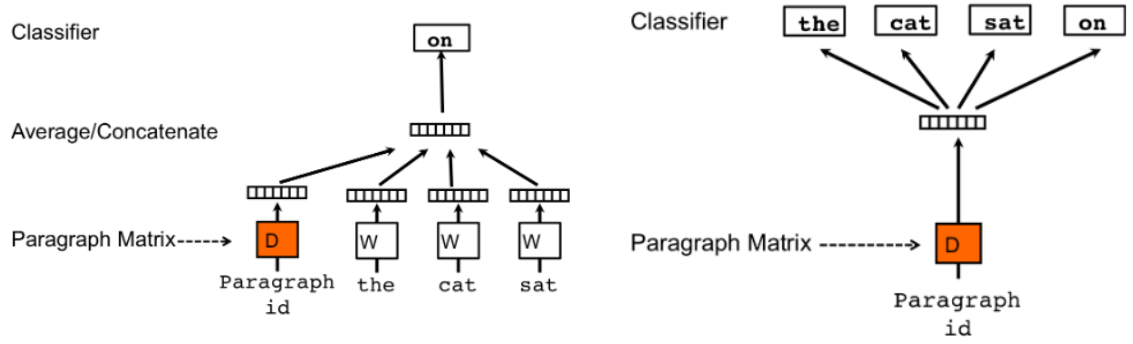
$$(tf\_idf)_{t,d} = \frac{\#\ de\ "t"\ dans\ "d"}{\#\ de\ terme\ dans\ le\ document} \times \log \frac{\#\ de\ documents}{\#\ de\ documents\ avec\ le\ terme\ t} \tag{1}$$

	Abyde	ville	Egypte
01-202	0	0	0,079
01-349	0,207	0,027	0
01-350	0.829	0,110	0,318

Table 5: Exemple de représentation avec la technique TF-IDF

L'avantage de la méthode est au niveau lexical. En revanche, ce type de représentation ne capture pas le contexte, ou l'effet de la position des mots, en d'autres termes les documents avec les mêmes mots seront identiques.

Doc2Vec est une technique de plongement de texte basée sur la technique de plongement de mots Word2Vec [Le and Mikolov, 2014]. Elle fonctionne de manière presque identique à celle-ci. En plus de la matrice des vecteurs de mots, Doc2Vec va générer une matrice de vecteurs de documents. Un vecteur document portera dès lors le sens du document auquel il est associé et celui-ci est généré en même temps que les vecteurs des mots constituant le document.



(a) Doc2Vec PV-DM

(b) Doc2Vec PV-DBOW

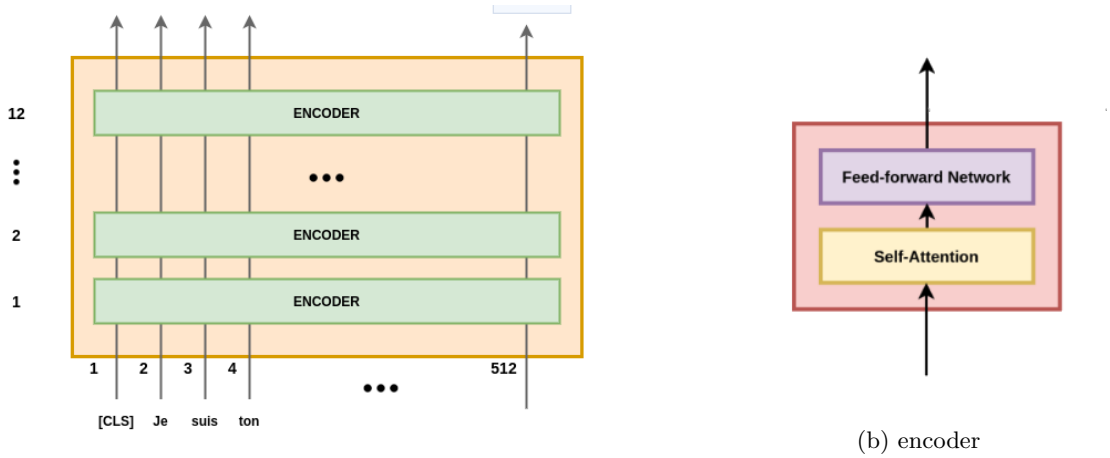
Figure 8: Doc2Vec [Le and Mikolov, 2014]

Par conséquent, le réseau de neurones du modèle Doc2Vec fera donc une prédiction d'un mot, dans le cas Doc2Vec PV-DM (voir figure 8a), en sachant son contexte (les mots l'avoisinant) et le document, et respectivement le cas Doc2Vec PV-DBOW (voir figure 8b), le réseau de neurone prédit un échantillon de mots tiré d'une fenêtre fixé au préalable, seulement en connaissance du vecteur document. Ainsi dans tous les cas, le vecteur document sera lui aussi mis à jour et représentera à la fin un vecteur dense du document exploitable pour la classification. Cette représentation pallie les contraintes des deux précédentes c'est-à-dire la taille des vecteurs et l'expression du contexte dans la représentation.

FastText [Bojanowski et al., 2017] est aussi une extension de Word2Vec. Elle essaye d'améliorer la technique en prenant en compte l'information de n-grammes de caractères (sous-mots), c'est à dire au lieu d'apprendre directement les représentations de mots, on utilise plutôt les n-grammes des mots. On prend comme exemple le mot 'apprendre' et n fixé à 3, on obtient les n-grammes suivants : "app", "pre", "ren", "end", "dre". Pour obtenir la représentation de ce dernier mot, un modèle Word2Vec est entraîné afin d'obtenir les représentations de ses n-grammes, ainsi la représentation du mot sera la somme des vecteurs de ses n-grammes. L'avantage du modèle FastText est que les représentations engendrées sont significatives malgré la rare apparition de certains mots dans le corpus d'entraînement voire des mots qui n'apparaissent pas du tout "out-of-vocabulary". FastText sera utilisé pour avoir les plongements des mots. Par la suite, ces plongements seront utilisés l'initialisation des couches *embeddings* des réseaux de neurones pour la classification.

Bert ou *Bidirectional Encoder Representations from Transformers* [Devlin et al., 2018] est un modèle de représentation de textes et qui a la particularité d'être contextuel c'est à dire que la représentation vectorielle d'un mot dépendra du contexte du mot dans un texte. L'interception du contexte est bidirectionnelle contrairement à d'autres architectures de modèle, et donc le plongement d'un mot prend en compte les mots précédents et ceux qui suivent ce dernier.

Bert utilise les réseaux Transformers [Vaswani et al., 2017] (figure 9a). C'est un étagement de 12 couches



(a) transformers

(b) encoder

Figure 9: Architectures utilisées par Bert

d'*encoders* ayant la même structure mais différents poids. Tel est l'avantage de Bert, il utilise l'attention sur la séquence (figure 9b) au lieu de connexions récurrentes, cela permet de traiter des relations entre mots très éloignés dans la séquence.

Le modèle Bert est un modèle pré-entraîné sur différentes données de textes bruts, par exemple: Wikipédia (2500 millions de mots), textes de livres (800 millions de mots). Le pré-entraînement est réalisé sur deux tâches différentes, la première est *Masked Language Modeling* qui consiste à prédire le mot suivant en sachant le début de la phrase. La deuxième tâche est le *Next Sentence Prediction*, il s'agit de prédire si une séquence est oui ou non la suite d'une autre séquence. Ce modèle comprend 512 entrées qui correspondent au nombre de *tokens* que peut prendre le modèle en entrée, le premier token est "[CLS]" qui équivaut au début de la séquence. En sortie, le modèle comprend également 512 sorties de tailles 768 et qui correspondent aux vecteurs représentatifs des *tokens* sauf le premier vecteur qui est la représentation vectorielle de toute la séquence ou le document en entrée et qui, dans notre cas, sera utilisé pour la classification.

DistilBert, Camembert & Flaubert Pour la classification, on a aussi exploité des variantes du modèle Bert. La première est DistilBert [Sanh et al., 2019] qui est une version allégé de Bert qu'on utilisera pour avoir une exécution plus rapide, et un modèle plus léger. Nous avons comparé les deux modèles pour étudier leurs performances.

Camembert [Martin et al., 2019] et Flaubert [Le et al., 2019] sont des variantes pré-entraînées pour la langue française. Plus précisément, le modèle Camembert a été entraîné sur le corpus oscar[8] extrait de Common Crawl [9], quant à Flaubert, il est pré-entraîné sur des données dont la source est presque similaire comme Wikipédia, livres et d'autres textes français extraits de plusieurs sources. Nous avons également souhaité comparer ces modèles avec les précédents afin d'observer les différences de performances.

MUSE Embeddings MUSE ou Multilingual Universal Sentence Encoder [Yang et al., 2019] est dérivée de USE (Universale Sentence Encoder) [Cer et al., 2018] c'est un modèle d'encodage de phrases multilingue qui tient compte de 16 langues différentes dont le français. L'objectif de l'entraînement de ce modèle est d'avoir une représentation de textes, tirés de langues différentes, dans un même espace vectorielle, en utilisant une

[8] https://oscar-corpus.com/
[9] https://commoncrawl.org/

architecture d'apprentissage profond particulière " Multi-task Dual Encoder" [Chidambaram et al., 2018] qui vise à faire un entraînement du modèle sur plusieurs tâches : une tâche de prédiction de questions-réponses à fonctionnalités multiples, une tâche de classement de traductions et une tâche d'inférence en langage naturel. On obtient grâce à cet encodeur une représentation de taille 512 pour chaque document.

4.2.4 Classification

Pour la partie classification, plusieurs méthodes ont été utilisées, allant des méthodes classiques aux méthodes d'apprentissage profond et au raffinage des modèles pré-entraînés.

Classifieurs classiques Afin de procéder à la classification, l'utilisation des méthodes classiques permet d'obtenir des scores de références avec des temps d'entraînement relativement courts qui s'avèrent une référence notamment grâce à la particularité mathématiques de chacune ou probabiliste comme le classifieur Naïve Bayesienne Multinomial. Dans notre cas, il s'agit d'une étape de classification supervisée. Elle prend en entrée le contenu des articles vectorisé (selon les approches décrite dans la section précédente) ainsi que les labels correspondants, c'est-à-dire la classe que le modèle devrait prédire pour ces documents. Nous avons souhaité tester plusieurs algorithmes de classification supervisé, tels que la méthode Naïve Bayesienne Multinomial. Cette méthode probabiliste n'est cependant utilisable qu'avec les représentations vectorielles statiques comme le sac de mots et le TF-IDF car celles-ci ne comprennent que des valeurs positives. Nous avons également testé d'autres algorithmes tels que les arbres de décisions, la régression logistique, le gradient de descente stochastique, les forêts d'arbres décisionnels, les K plus proches voisins (KNN) et les Machine à vecteurs de support (SVM). Les résultats sont présentés dans la section 5.

Classifieurs profond Afin d'examiner l'apport en amélioration des réseaux de neurones dans la tâche de classification, on a opté pour une variété de réseaux de neurones et de combinaisons de techniques listées ci-dessous:

- Multilayer Perceptron (MLP) appliqué sur les représentations vectorielles citée dans la section 4.2.3
- Réseaux de neurones profond de type ANN (Artificial Neural Network) mais aussi LSTM (Long Short-Term Memory) et CNN (Convolutional Neural Network) avec ou sans une couche *embeddings*. Par rapport aux modèles avec la couche *embeddings* ils utilisent des *embeddings* initialisés aléatoirement puis entraînés au cours de la classification et également des *embeddings* pré-entraînés comme FastText.

Classifieurs pré-entraînés Il est également intéressent d'exploiter les classifieurs pré-entraînés comme par exemple : Bert avec une méthode de *Fine-tuning*. L'objectif est de prendre les poids d'un réseau de neurones pré-entraîné et de l'utiliser comme initialisation dans un nouveau modèle et sur d'autres données spécifiques. On utilise dans ce cas de figure un modèle bien spécifique pour la classification nommé "BertForSequenceClassification", c'est un modèle Bert comme décrit dans la sous-section 4.2.3 avec une couche linéaire par dessus qui se charge de faire la classification (figure 10).

Ce dernier modèle sera désormais entraîné sur nos données, ainsi à mesure que le modèle est alimenté par les données, les poids du classifieur et du modèle Bert sont mis à jours. Un détail important dans ce genre d'application est de faire le formatage nécessaire aux données avant d'alimenter le modèle, comme il est indiqué sur les premières couches du modèle de la figure 10, les données sont d'abord tokenisées. Des *tokens* spéciaux sont ajoutés aux séquences pour indiquer le début et la fin de la séquence et par la suite récupérer la représentation vectorielle de la séquence entière, et enfin un *padding* est effectué pour que le modèle ait des entrées de taille fixe, parallèlement un masque est aussi construit afin de différencier les vrais

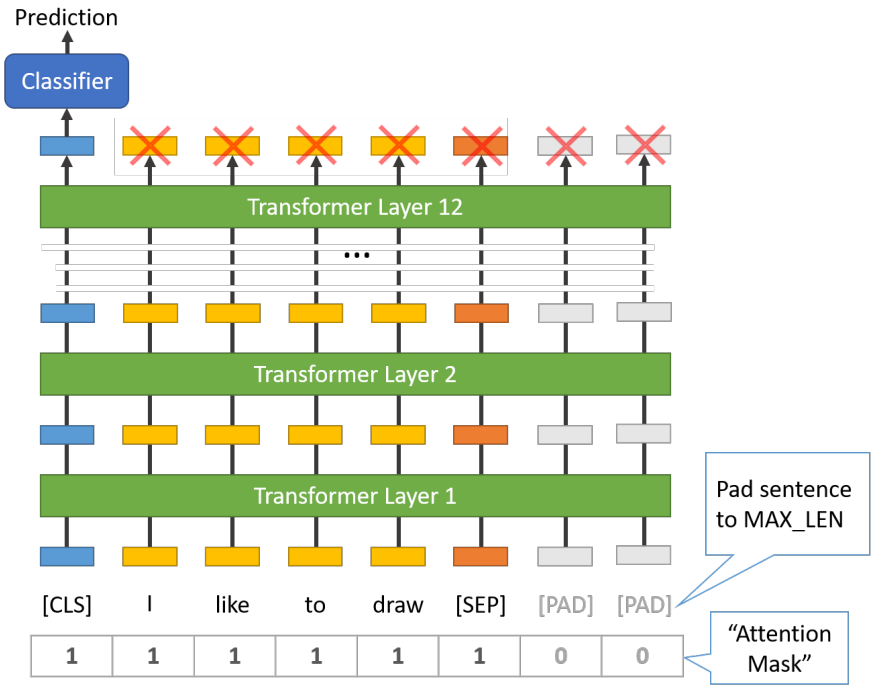


Figure 10: Modèle BertForTextClassification

tokens de la séquence à celle du *padding*. Par ailleurs, on observe dans la figure que le classifieur est lié à la première sortie qui correspond au *token* "[CLS]" car celui ci correspond à la représentation de tout le texte.

Il est intéressent d'utiliser également le *Fine-tuning* des variantes de Bert, sachant que Bert est un modèle Multilingue, alors que Camembert et Flaubert ont été entraînés pour la langue française et peuvent donc produire des résultats meilleurs.

4.3 Modules Additionnels

Dans le but d'améliorer les résultats de la classification, une étape postérieure à l'entraînement de classifieur est réalisée, celle-ci réside dans la recherche des bons hyper-paramètres des modèles.

D'un autre côté, on a réalisé plusieurs démarches pouvant être susceptibles d'apporter de meilleurs résultats comme l'utilisation des architectures de réseaux profonds complexes comme le Modèles Hiérarchique d'Attention ou les réseaux convolutifs sur graphe, ou d'autres méthodes qui traitent au niveau de l'étape de l'extraction des caractéristiques.

4.3.1 Modèle Hiérarchique d'attention

Ce modèle hiérarchique part du principe que les mots constituent les phrases et les phrases constituent le document, c'est à dire que le modèle cherche à capturer le sens des documents à partir des phrases et le sens des phrases à partir des mots, mais comme les mots d'une phrase n'ont pas la même importance ainsi que les phrases au sein d'un texte, le mécanisme d'attention est utilisé alors dans cette architecture.

Comme le montre la figure 11, le réseau HAN est constitué de deux couches principales, l'une pour le traitement au niveau des mots et l'autre au niveau des phrases. Les RNN bidirectionnels et les couches d'attention sont les deux sections du modèle d'attention et qui sont présents dans les deux couches. Alors

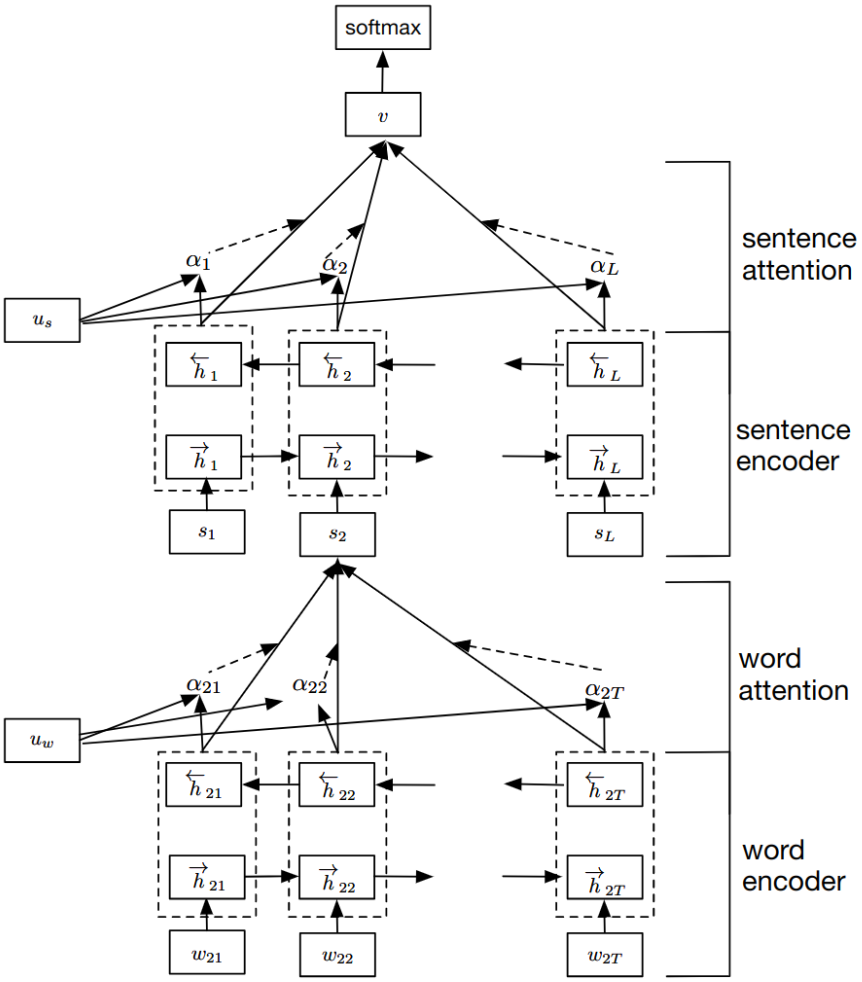


Figure 11: Réseaux d'attention hiérarchiques (HAN) [Yang et al., 2016]

qu'un RNN bidirectionnel apprend la signification d'une séquence de mots, capture le sens de façon bidirectionnelle puis produit un vecteur pour chaque mot, le mécanisme d'attention utilise son propre réseau neuronal peu profond pour obtenir des poids pour chaque vecteur de mot. Ensuite, il fait une somme pondérée de chaque vecteur mots avec son poids adéquat pour obtenir la représentation de la phrase. La même méthode est utilisée pour les vecteurs de phrases au niveau de la couche supérieure des phrases, ce qui donne un vecteur final qui encapsule le document. Une dernière couche est alors disposée sur le réseau pour la classification. L'intérêt de ce réseau d'attention hiérarchique réside dans son appui sur l'importance des mots dans les phrases et des phrases dans le document.

4.3.2 Réseaux convolutifs de graphes

Les méthodes de représentation de texte sont nombreuses, beaucoup se basent sur les RNN et CNN. Elles capturent la sémantique ou l'information structurelle de manière locale. Il existe une méthode à base des GCN: réseaux convolutifs de graphes [Yao et al., 2019] qui se rapporte aux co-occurences des mots pour capturer l'information globale du corpus. La classification de texte à base de réseaux convolutifs de graphe

a surpassé beaucoup de classifieurs ce qui suscite l'intérêt de la tester sur notre corpus.

Cette méthode commence par construire un graphe à partir du corpus. Les documents et mots du corpus sont des noeuds du graphe et les liens entre ces noeuds vont constituer des poids. Un poids entre un noeud document et un noeud mot est la valeur TF-IDF de ce mot dans ce document. Un poids entre un mot i est un mot j est la valeur PMI (Pointwise Mutual Information) des deux mots décrite par les équations ci-dessous.

W est une fenêtre coulissante de mots, c'est à dire #W(i,j) est le nombre de fenêtres coulissantes où le mot i et le mot j apparaissent. #W est le nombre de fenêtres coulissantes dans le corpus.

$$PMI(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{2}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{3}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{4}$$

Une fois ce graphe construit, nous l'alimentons à notre réseau à base de deux couches GCN. Ce réseau agit sur le graphe comme les CNN agissent sur les images, il fait l'apprentissage des poids sur les noeuds du graphe. La sortie finale est ensuite introduite dans une couche softmax avec une fonction de perte d'entropie croisée pour la classification.

4.3.3 Sélection des caractéristiques

La sélection des caractéristique est le procédé de sélectionner les mots ou phrases les plus pertinents et cela dans le but d'améliorer les résultats de la classification ainsi que la complexité à travers la réduction de dimension. Pour résoudre le problème, on opte pour l'utilisation d'une méta-heuristique qui est l'algorithme génétique.

Pour développer cet algorithme, il faut avoir au préalable des initialisations et des fonctions qu'on utilisera comme la fonction d'évaluation. C'est pour ce motif qu'on exploite une version de l'algorithme génétique présenté dans [Ghareb et al., 2016]. Nous pouvons, en revanche, choisir le type de vectorisation et le classifieur dont on souhaite améliorer les résultats.

Les étapes de l'algorithme sont représentés dans la figure 12 et les détails de ces derniers sont comme suit :

- Population Initiale: Dans cette étape, il est question de générer des solutions possibles. Une solution est un sous-ensemble de termes extraits à partir des termes du vocabulaire. Les solutions peuvent se chevaucher, c'est à dire, l'intersection des solutions n'est pas un ensemble vide, ils peuvent des termes en commun. Le point important dans cette étape et la façon d'encoder la solution. Dans ce cas de figure, une solution est un vecteur de 1 et de 0, où 1 à la position i signifie que le i-éme terme du vocabulaire est présent dans la solution, 0 si ce terme ne l'est pas.

- Évaluation: On parle de fonction de fitness, qui est un hyper-paramètre de l'algorithme et qui permet d'évaluer la performance des solutions générées dans l'étape précédente, en d'autres terme, la performance des sous-ensembles de termes. Par la suite, les meilleurs sous-ensembles seront utilisés pour générer les prochaines solutions possibles. Nous choisirons la fonction de fitness de [Ghareb et al., 2016] dont l'équation est la suivante: $FitnessFunction(S_i) = Z \times C(S_i) + (1 - -Z)(1/Size(S_i))$, où S_i est un sous-ensemble de terme, $Size(S_i)$ est la taille du sous-ensemble, $C(S_i)$ est la f-mesure de la classification avec les termes du sous-ensemble S_i et enfin Z est un paramètre compris en 0 et 1. La valeur de la

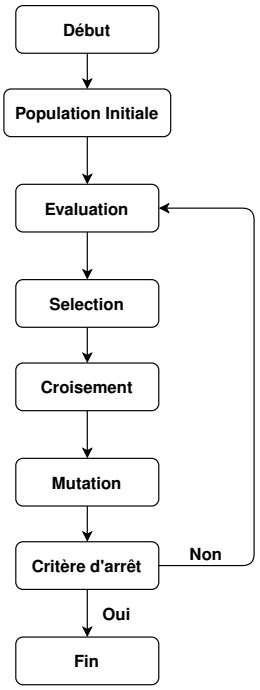


Figure 12: Étapes de l'algorithme génétique

fonction fitness augmente si le resultat f-mesure de la solution S_i et diminue si la taille S_i augmente. Si Z est grand alors on privilégie la performance f-mesure obtenue par rapport à la taille du sous-ensemble de la solution. La valeur de Z peut être fixée à 80%.

- Sélection: Elle consiste à choisir des solutions pertinentes à partir desquelles on va générer d'autres solutions. Il existe pour cela plusieurs stratégies, nous adopterons une stratégie dite Roulette Wheel Selection [Lipowski and Lipowska, 2012]. Chaque sous-ensemble à une probabilité d'être sélectionné qui est en fonction de la fonction fitness comme l'indique l'équation suivante $p_i = \frac{FitnessFunction_i}{\sum_{j=1}^{N} FitnessFunction_j}$, où N est le nombre de solutions. Donc, si une solution a un score élevé elle aura plus de chance d'être sélectionnée. Avec ces probabilités, nous allons construire des intervalles de probabilités où chaque intervalle correspond à une solution et les solutions dont le score est élevé sont plus grands. On génère des nombres aléatoires R entre 0 et 1, on prend par la suite le sous-ensemble dont l'intervalle correspond au R généré. Ce sous-ensemble est alors utilisé pour la prochaine étape qui est le croisement.

- Croisement: Cette étape sert à générer une nouvelle population de solutions. À partir des solutions sélectionnées dans l'étape précédentes, on prend des solution ou sous-ensemble deux à deux dans l'ordre de sélection. On découpe chaque sous ensemble en deux parties de la même taille. Puis, on calcule pour chaque partie obtenue les poids cumulatives de ces termes où le poids représente la valeur TF-IDF, par exemple, calculée auparavant lors de l'étape de l'extraction des caractéristiques. Enfin, on concatène les deux parties ayant obtenues le meilleur cumul de poids, et on concatène aussi les deux parties restantes pour la diversité des solutions.

  En procédant de cette façon sur l'ensemble de la population, nous obtenons un nouveau ensemble de population.

- Mutation: L'étape de mutation consiste à avoir une diversité dans la solution selon plusieurs stratégies.

La stratégie choisie est simple, à partir de la nouvelle population générée, on omet quelques termes et on rajoutent d'autres, en changeant quelques 0 par des 1 et vise versa dans la codification dans la solution. On obtient, dans ce cas, une nouvelle population. À compter de cette étape, nous vérifions le critère d'arrêt, si un nombre d'itération fixé au début est atteint ou si le score vaut une valeur fixée également nous arrêtons le processus et nous récupérons le sous-ensemble ayant donnée le meilleur résultat sinon on réitère à l'étape évaluation en prenant comme la population initiale, la population produite dans l'étape Mutation.

5 Expérimentations et résultats

5.1 Outils de développement

Dans le but de la réalisation de notre solution, le choix des technologies s'est porté sur les suivantes:

Python est un langage de programmation *opensource*, multiparadigme et multiplateformes. Il est simple et très facile à coder avec, il est très utilisé dans les projets de science des données et de l'apprentissage automatique, ce qui convient au projet, notamment grâce à ses nombreuses librairies.

Pytorch est une bibliothèque Python développée par Facebook, pour l'apprentissage profond. Elle est facile à apprendre et à manipuler, elle est utilisée pour implémenter certains modèles proposés dans notre solution.

BeautifulSoup est une librairie Python pour faire l'extraction des données à partir de fichier XML et HTML. Elle est utilisée pour extraire les données des fichiers XML-TEI fournit par l'ARTFL.

NLTK est une bibliothèque Python permettant d'effectuer différentes tâches de traitement automatique du langage telles que la tokenisation, la lemmatisation et la suppression des mots vides. Elle est utilisée pour la préparation et le pré-traitement des données avant l'étape de vectorisation.

Huggingface est une communauté spécialisée dans le TAL qui a développé la librairie Hugginface[10]. Cette librairie comprend plusieurs classes pour la réalisation de multiples tâches d'intelligence artificiel. Nous utilisons l'implémentation de Bert faite dans cette librairie.

Keras Keras est également une bibliothèque Python implémentant des modèles à base de réseaux profonds et que nous utilisons pour l'implémentation de certains classifieurs profonds.

Google Colaboratory est un service de Google pour travailler en collaborations sur des projets exécutés à distance sur les serveur de Google. Il offre un environnement puissant en ressource, et des outils comme Jupyter Notebook ne nécessitant aucune configuration, et des bibliothèques Python dédiés aux projets d'apprentissage automatique.

[10] `http://huggingface.co`

Git est un outil libre pour la gestion et le versionning de code principalement ainsi que de collaboration et revue de code. Il est très utile dans de grands projets avec plusieurs développeurs pour une bonne gestion de code. Nous utilisons une instance de Gitlab hébergée au LIRIS pour partager le code développé dans le cadre de ce stage[11].

5.2 Expérimentations

Comme nous l'avons décrit dans les sections précédentes nous avons souhaité tester et comparer différentes approches de classification. Les différentes expérimentations menées sont décrites dans la figure 13. Le processus se décompose en deux étapes principales : la vectorisation et la classification supervisée. Nos expérimentations reposent sur la combinaison des approches de vectorisation avec les algorithmes de classification. Plusieurs configurations sont étudiées :

- Une vectorisation classique avec les techniques de sac de mots ou de TF-IDF associée à des méthodes de classification supervisée (Naive Bayes, Logistic regression, Random Forest, SGD et SVM).
- Une vectorisation par une technique de plongement de mots statique (Doc2Vec) associé aux mêmes méthodes de classification que précédemment (à l'exception de Naive Bayes).
- Une vectorisation par une technique de plongement de mots dynamique (BERT, DistilBERT, CamemBERT) également associée aux mêmes algorithmes de classification que précédemment
- Une technique de classification 'end-to-end' utilisant un modèle de langue pré-entraîné et une technique de *fine-tuning* pour ré-entraîner un modèle de classification spécifique.

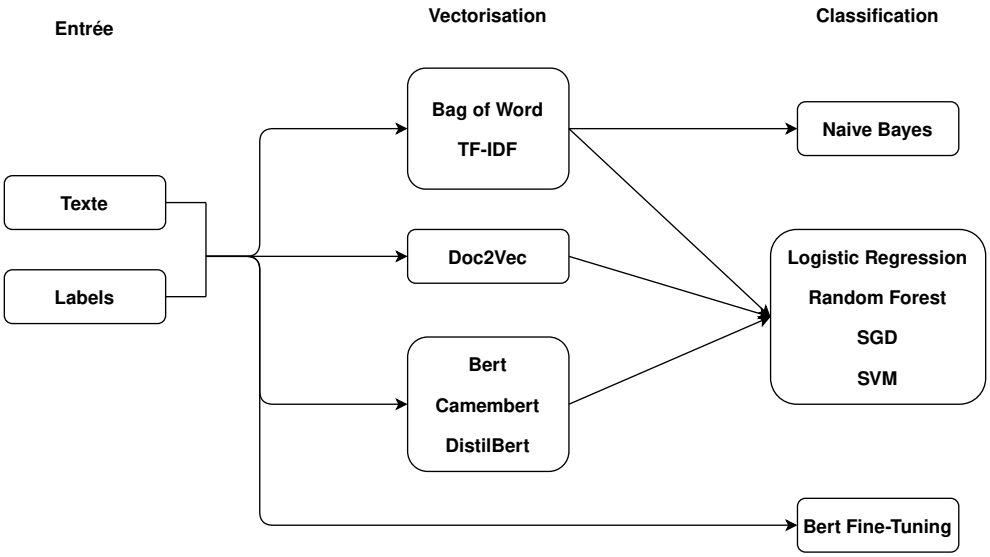


Figure 13: Schéma des expérimentations

5.3 Jeu de données

Le jeu de données utilisé pour l'entraînement et l'évaluation de la classification est constitué des articles catégorisés de l'Encyclopédie. Les articles non classés sont mis de côté et seront classés par la suite grâce aux modèles entraînés.

[11] `https://gitlab.liris.cnrs.fr/geode/EDdA-Classification`

Source	# de classes	# d'articles non classés
ARTFL	1 654	12 685
ENCCRE	44	2 392

Table 6: Caractéristique du jeu de données

Nous établirons un découpage des données en deux ensembles, un ensemble de jeu pour l'entraînement et un autre pour l'évaluation. Cependant, comme les classes sont déséquilibrées, nous utilisons le concept de stratification. La stratification permet d'avoir une répartition équitable des classes entre les deux ensembles d'entraînement et de validation et donc on évite que les classes minoritaires ne participent pas à l'évaluation ce qui engendre des résultats erronés.

5.4 Métriques

Pour l'évaluation des modèles entraînés nous utilisons les métriques de classification suivantes :

- Précision: c'est une métrique de classification qui sert à répondre à la question : "Quelle proportion identifiée positive est correcte ?"

$$Precision = \frac{VraisPositifs}{VraisPositifs + FauxPositifs} \tag{5}$$

- Rappel: dite aussi la sensiblité, le rappel est la fraction d'instances pertinentes trouvées parmi toutes les instances pertinentes.

$$Rappel = \frac{VraisPositifs}{VraisPositifs + FauxNegatifs} \tag{6}$$

- f-mesure: Cette mesure est une moyenne qui prend en compte les deux mesures précédentes, d'où la formule :

$$F - mesure = \frac{2 \times Rappel \times Precision}{Rappel + Precision} \tag{7}$$

Il est à noter que lors de la classification, nos classes ne sont pas toutes équilibrées. Malgré le rééquilibrage fait, il y aurait certains cas où ce problème n'est pas corrigé, donc nous nous fierons aux métriques citées précédemment mais pondérées par le nombre d'instances par classe.

5.5 Résultats

Nous présentons dans cette section les résultats obtenus lors de l'expérimentation des différentes techniques de classification citée dans la section précédente. Il est important de bien comprendre ces résultats et les analyser pour améliorer les performances des classifieurs.

Le tableau 7 présente les f-mesures des classifieurs classiques avec les techniques vectorisation Sac de mots, TF-IDF et DOC2Vec. Ces résultats sont obtenus pour une classification selon les 38 ensembles de domaines ainsi que selon les 143 domaines proposés par ENCCRE. Les colonnes 50-500 et 50-1500 expriment les valeurs des hyper-paramètres qui concerne le rééquilibrage des classes : 50 est le nombre minimum d'instances qu'une classe doit avoir, donc on aura des classes bien représentées, quant aux valeurs 500 et 1500 correspondent au nombre maximum d'instances dans une classe, cela permet d'avoir la différence entre le nombre d'instances entre les classes pas très grande.

Classifieur	Vectorisation	Ensemble domaines (38)		Domaines (143)	
		50-500	50-1500	50-500	50-1500
Naive Bayes	Bag of Words	0.55	0.58	0.43	0.44
	TF-IDF	0.60	0.57	0.44	0.40
Logistic Regression	Bag of Words	0.58	0.60	0.47	0.49
	TF-IDF	0.62	0.62	0.52	0.53
	Doc2Vec	0.54	0.57	-	0.49
Random Forest	Bag of Words	0.51	0.48	0.36	0.33
	TF-IDF	0.52	0.47	0.36	0.34
	Doc2Vec	0.47	0.48	-	0.36
SGD	Bag of Words	0.61	0.62	0.52	0.54
	TF-IDF	0.65	0.64	0.56	0.56
	Doc2Vec	0.55	0.58	-	0.51
SVM	Bag of Words	0.53	0.55	0.42	0.45
	TF-IDF	0.61	0.63	0.51	0.53
	Doc2Vec	0.50	0.56	-	0.50

Table 7: Résultats f1-mesure des classifieurs classiques

On remarque que les résultats sont plutôt moyens (entre 50 et 60%), le classifieur SGD se démarque des autres classifieurs avec une f-mesure de 0.65 pour une vectorisation tf-idf, et 0.61 avec une représentation sac de mots. De la même manière, les classifieurs SVM et régression logistiques ont des f-mesures très proches aux environs de 0.61. Les résultats sont moins bons quand le nombre de classes accroît, ceci est dû au fait que les classes deviennent mal représentées, que le déséquilibres entre elles augmente et que la frontière lexicale ou sémantique entre les classes devient plus compliqué à déterminer.

Ensemble domaines	F1	Ensemble domaines	F1	Ensemble domaines	F1
Agriculture - Economie rustique	0,68	Economie domestique	0,38	Militaire (Art) - Guerre - Arme	0,74
Anatomie	0,80	Géographie	0,87	Minéralogie	0,47
Antiquité	0,64	Grammaire	0,43	Monnaie	0,65
Architecture	0,72	Histoire	0,42	Musique	0,80
Arts et métiers	0,32	Histoire naturelle	0,62	Pêche	0,75
Beaux-arts	0,65	Jeu	0,84	Pharmacie	0,30
Belles-lettres - Poésie	0,48	Maréchalerie - Manège	0,78	Philosophie	0,45
Blason	0,87	Marine	0,73	Physique	0,65
Caractères	0,70	Mathématiques	0,71	Politique	0,48
Chasse	0,83	Médailles	0,43	Religion	0,72
Chimie	0,49	Médecine - Chirurgie	0,58	Spectacle	0,14
Commerce	0,63	Mesure	0,42	Superstition	0,39
Droit - Jurisprudence	0,68	Métiers	0,54		

Table 8: Ensemble domaines, SGD (50-1500)

Cette disparité entre les classes est visible dans le tableau 8. On peut apercevoir que certaines classes ont une très bonne f-mesure comme les classes 'Anatomie', 'Architecture', 'Blason' et 'Géographie' qui est

de 0.87, 'Musique' et 'Jeu' qui est de 0.80 et 0.84 respectivement. On en conclut que le résultat général est moyen à cause de la disposition de quelques classes comme 'Spectacle' qui sont mal représentée par leur vocabulaire ou avec un nombre d'articles faible, comme le cas de la classe 'Spectacle' qui comprend que 88 instances. Par ailleurs la proximité lexicale ou sémantique entre les différentes classes est aussi visible sur les matrices de confusion (figure 14).

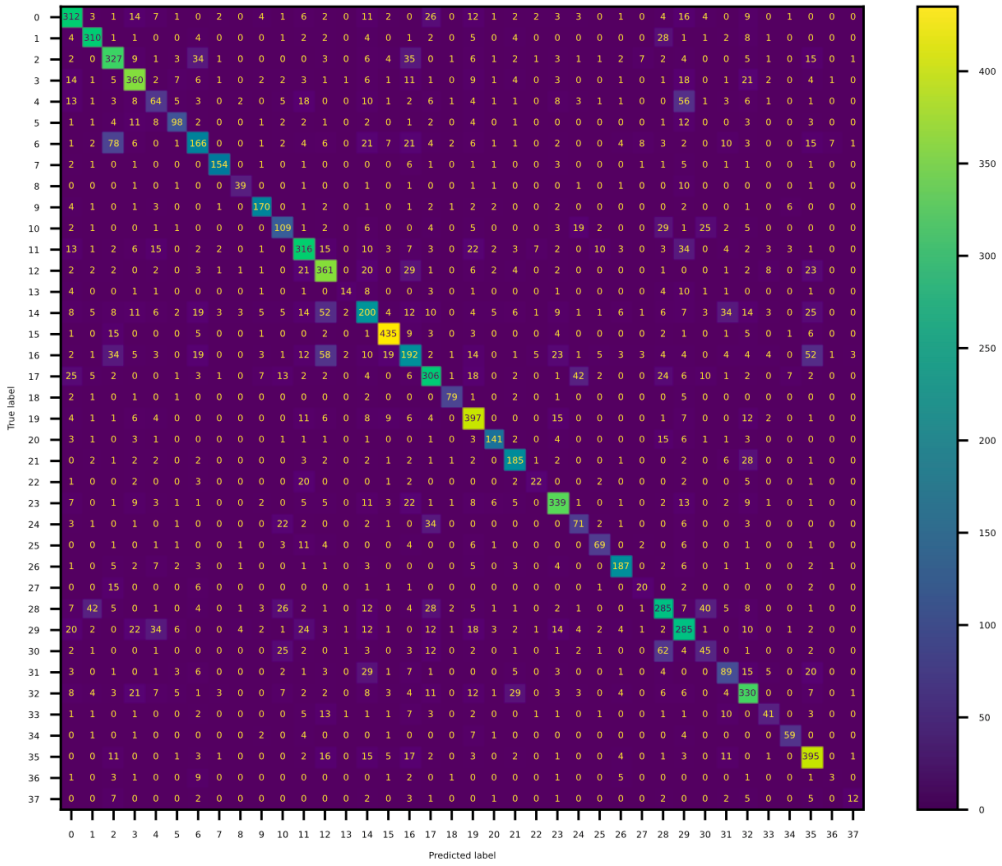


Figure 14: Matrice de confusion : Ensemble domaines, SGD (50-1500)

La figure 14 représente la matrice de confusion correspondant au résultat précédent du tableau 8. On remarque une diagonale qui se distingue indiquant que plusieurs classes sont bien classées. Néanmoins, il y a des éparpillements qui reflètent une mauvaise classification. Les valeurs de cette matrice nous permet d'identifier des confusions entre certaines classes, par exemple 78 articles de la classe 'belles-lettres' ont été classés par le modèle dans la classe 'antiquité'. De la même manière 62 articles de pharmacie sont classés en médecine. Cela montre bien la difficulté de segmenter et de différencier certaines classes tant d'un point de vue lexicale que sémantique.

Classifieur	Vectorisation	F-mesure (articles complets)	F-mesure (1er paragraphe)
Logistic Regression	Bert	0.3894	0.416
	Camembert	0.581	0.562
	DistilBert	0.436	0.432
Random Forest	Bert	0.336	0.308
	Camembert	0.335	0.491
	DistilBert	0.326	0.326
SGD	Bert	0.387	0.371
	Camembert	0.549	0.527
	DistilBert	0.382	0.379
SVM	Bert	0.377	0.376
	Camembert	0.54	0.557
	DistilBert	0.390	0.373

Table 9: Ensemble domaines 50-100

Le tableau 9 affiche les résultats F mesure obtenus avec les vectorisations Bert et ses variantes appliqués sur un corpus constitué d'articles complet et un autre corpus constitué que du premier paragraphe de chaque article.

On constate que le modèle Camembert accomplit de meilleurs résultats que les autres modèles. Ceci est dû au fait que Camembert est conçu pour le traitement des corpus en français. La médiocrité des résultats est probablement à cause de la taille des données étant donné qu'un échantillon de 100 instances est pris de chaque classe, ce qui est très peu pour ces modèles. Les modèles Bert sont très gourmands en ressources, cette contrainte a donné lieu à ce choix de paramètres. Enfin, on relève de ce tableau que les résultats produits par les deux corpus sont semblables, c'est pourquoi on peut exploiter les paragraphes des articles pour gagner en coût de traitement étant donné la différence de volume entre les deux corpus.

6 Conclusion

Le travail réalisé lors de ce stage a traité une problématique très intéressante de classification d'articles encyclopédiques. La finalité de ce travail est de réaliser un classifieur capable d'identifier les classes des articles, c'est-à-dire d'apprendre à partir des représentations des documents les schémas et caractéristiques afin de déterminer la classe des articles non classés de l'Encyclopédie De Diderot et d'Alembert. Par ailleurs, dans les perspectives du stage nous nous intéresserons à la classification des articles issus d'autres encyclopédies françaises.

Pour la résolution de cette problématique, nous avons enchaîné sur des étapes logiques et bien définies allant de l'extraction des données à leur pré-traitement puis on a exploité différentes méthodes pour la représentation des documents afin d'entraîner plusieurs modèles avec différents algorithmes ayant chacun ses spécificités.

Les résultats obtenus sont encourageants mais nécessitent une amélioration. La représentation de certaines classes ainsi que le déséquilibre sont les points évidents qui causent une marge négative des scores. Cependant, certaines classes sont bien repérées et en particulier des classes très intéressantes dans le cadre du projet GEODE telles que la classe 'Géographie'. Les résultats obtenus sont ainsi directement exploitables pour la poursuite des autres objectifs et travaux en cours dans ce projet.

Nous avons observé que les modèles pré-entraînés comme Camembert produit de meilleurs résultats que

Bert qui est le modèle état de l'art actuel dans les tâches de TAL. On pourrait lors de futurs travaux exploiter cet avantage en sachant que le Fine-Tuning utilisé dans ce travail comprend une couche linéaire par dessus le modèle Camembert pour la classification et la simplicité de ce modèle pourrait être la raison d'un tel résultat. Il serait intéressent de construire des classifieurs plus développés et plus complexes afin d'observer si les résultats peuvent être supérieurs qu'avec les méthodes plus classiques d'apprentissage supervisé.

Enfin, les résultats pourraient également être améliorés au niveau du traitement des données, d'abord en explorant les techniques d'équilibrage des classes, puis en concevant un tokeniseur mieux adapté afin d'extirper un vocabulaire bien évident et enfin en explorant les techniques de sélection de caractéristiques qui sont très intéressantes car elles permettent de réduire la dimension des données et donc de réduire la complexité et également améliorer les résultats de la classification à travers la sélection des termes plus significatifs pour trouver des patterns pour la classification.

Ce stage a été pour moi une expérience très bénéfique qui m'a permis de progresser dans plusieurs aspects. D'abord, il a été très positif du point de vue technique j'ai acquis un savoir faire dans le domaine de la recherche étant donnée la nature du sujet qui est orientée recherche, j'ai aussi amélioré mes compétences techniques dans le domaine du Machine Learning et du NLP, où j'ai appris de l'expérience de mon encadrant Monsieur Ludovic MONCLA et de Madame Alice BRENON doctorante dans l'équipe DM2L et dont les conseils m'ont été très précieux. J'ai également beaucoup gagné en aptitude relationnelle grâce au suivi de la progression de mon stage grâce aux réunions hebdomadaires lors desquelles nous échangeons beaucoup sur le travail, cela m'a énormément aidé d'un point de vue communication et aussi en gestion de travail grâce à la régularité de cette communication qui m'a permis d'être rigoureux et travailler à un rythme soutenu.

Les difficultés rencontrées au cours de ce stage sont notamment des conséquences des circonstances sanitaires et aussi le travail à distance où il a fallu chercher la motivation mais une bonne organisation et des échanges constants à travers les réunions ont permis de pallier ces aléas et ont aidé au bon déroulement du stage.

Bibliographie

[Bojanowski et al., 2017] 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.

[Bromley et al., 1993] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[Chidambaram et al., 2018] Chidambaram, M., Yang, Y., Cer, D., Yuan, S., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*.

[Colas and Brazdil, 2006] Colas, F. and Brazdil, P. (2006). Comparison of svm and some older classification algorithms in text classification tasks. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 169–178. Springer.

[Conneau et al., 2016] Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

[Dada et al., 2019] Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., Ajibuwa, O. E., et al. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dieng et al., 2016] Dieng, A. B., Wang, C., Gao, J., and Paisley, J. (2016). Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

[Ghareb et al., 2016] Ghareb, A. S., Bakar, A. A., and Hamdan, A. R. (2016). Hybrid feature selection based on enhanced genetic algorithm for text categorization. *Expert Systems with Applications*, 49:31–47.

[Horton et al., 2009] Horton, R., Morrissey, R., Olsen, M., Roe, G., Voyer, R., et al. (2009). Mining eighteenth century ontologies: machine learning and knowledge classification in the encyclopédie. 3(2).

[Iyyer et al., 2015] Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691.

[Kowsari et al., 2019] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

[Le et al., 2019] Le, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., Lecouteux, B., Allauzen, A., Crabbé, B., Besacier, L., and Schwab, D. (2019). Flaubert: Unsupervised language model pre-training for french. *arXiv preprint arXiv:1912.05372*.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

[Li et al., 2020] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. (2020). A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364*.

[Lipowski and Lipowska, 2012] Lipowski, A. and Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196.

[Martin et al., 2019] Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de La Clergerie, É. V., Seddah, D., and Sagot, B. (2019). Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Peng et al., 2018] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., and Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pages 1063–1072.

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

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Roe et al., 2016] Roe, G., Gladstone, C., and Morrissey, R. (2016). Discourses and disciplines in the enlightenment: Topic modeling the french encyclopédie. *Frontiers in Digital Humanities*, 2:8.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Scarselli et al., 2008] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

[Shih et al., 2017] Shih, C.-H., Yan, B.-C., Liu, S.-H., and Chen, B. (2017). Investigating siamese lstm networks for text categorization. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 641–646. IEEE.

[Sreemathy and Balamurugan, 2012] Sreemathy, J. and Balamurugan, P. (2012). An efficient text classification using knn and naive bayesian. *International Journal on Computer Science and Engineering*, 4(3):392.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Weston et al., 2014] Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

[Xu, 2018] Xu, S. (2018). Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59.

[Yang et al., 2019] Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G. H., Yuan, S., Tar, C., Sung, Y.-H., et al. (2019). Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.

[Yang et al., 2016] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

[Yao et al., 2019] Yao, L., Mao, C., and Luo, Y. (2019). Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

[Yun-tao et al., 2005] Yun-tao, Z., Ling, G., and Yong-cheng, W. (2005). An improved tf-idf approach for text classification. *Journal of Zhejiang University-Science A*, 6(1):49–55.

[Zeng et al., 2018] Zeng, J., Li, J., Song, Y., Gao, C., Lyu, M. R., and King, I. (2018). Topic memory networks for short text classification. *arXiv preprint arXiv:1809.03664*.

[Zhang et al., 2010] Zhang, Y., Jin, R., and Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

[Zhang and Paramita, 2019] Zhang, Z. and Paramita, M. (2019). Product classification using microdata annotations. In *International Semantic Web Conference*, pages 716–732. Springer.

[Zhou et al., 2016] Zhou, X., Wan, X., and Xiao, J. (2016). Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 247–256.