Université de Strasbourg

UFR de mathématique et d'informatique Université de Strasbourg

**Hussam Ghanem**

Année universitaire 2019/2020

Master 2 Informatique: Sciences des données et des systèmes complexes

Rapport de stage de fin d'études :

# Word embeddings et deep learning pour la segmentation automatique de textes et l'extraction d'informations géographiques

LABEX IMU UNIVERSITÉ DE LYON

Ludovic MONCLA

Maître de conférences | INSA Lyon,

LIRIS CNRS UMR 5205

Équipe DM2L

ludovic.moncla@insa-lyon.fr

17/02/2020 - 15/07/202

# Remerciements

Ce rapport de stage cloture mon Master 2 à l'UFR MathInfo à l'Université de Strasbourg et également mon stage au sein de l'équipe DM2L (Data Mining et Machine Learning) au laboratoire LIRIS à l'INSA de Lyon. Je tiens à remercier sincèrement les personnes qui m'ont aidé dans la réalisation de ce projet.

Je souhaite adresser mes remerciements à Monsieur Ludovic MONCLA pour sa disponibilité et son aide tout au long de mon stage. Son soutien, ses encouragements, le temps qu'il a consacré pour m'accompagner dans tous les aspects du stage et ses conseils méthodologiques et techniques m'ont donné l'expérience dont j'avais besoin pour développer les méthodes et les solutions requises et m'ont permis d'acquérir de nombreuses nouvelles compétences. Je souhaite également remercier Jacques FIZE à l'examen de ce rapport, ce qui m'a sans nul doute permis d'en améliorer le contenu.

Le stage a été réalisé grâce au soutien financier du LABEX IMU (ANR-10-LABX-0088) de l'Université de Lyon, dans le cadre du programme « Investissements d'Avenir » (ANR-16-IDEX-0005) géré par l'Agence Nationale de la Recherche (ANR)

# Résumé

L'application des méthodes d'extraction automatique de données à partir de textes tels que les romans pose différents défis liés à l'hétérogénéité des documents et à l'ambiguïté de la langue. Pour répondre à cette problématique, nous proposons de développer une méthode de classification de texte qui aide à isoler les parties homogènes d'un texte pour y appliquer ensuite des méthodes d'extraction d'informations. Parmi les modèles récents de classification de texte qui utilisent les word embeddings, nous utilisons le modèle pré-entraîné BERT en appliquant la méthode de *Fine-Tuning*. Nous proposons une comparaison de notre méthode avec d'autres méthodes d'apprentissage classiques. Nous obtenons de bons résultats avec environ 97% d'exactitude et jusqu'à 100% de précision. La limitation rencontrée est que l'évaluation de cette méthode sur des corpus ayants des structures différentes de celle du corpus d'entraînement réduit la performance. Nous mettons les notebooks[1] Jupyter avec le code et l'ensemble des données d'entraînement et d'évaluation à disposition de la communauté.

[1] https://gitlab.liris.cnrs.fr/HExtGEO/sentence-classification

# Introduction

Au cours des cinq derniers mois, j'ai effectué mon stage au sein de l'équipe DM2L (*Data Mining* et *Machine Learning*) au laboratoire LIRIS (Laboratoire d'Informatique en Image et Systèmes d'Information) à l'INSA de Lyon. Au cours de cette période, j'ai eu l'occasion de perfectionner mes compétences en apprentissage automatique, en apprentissage profond et en traitement automatique de langage (TAL). Cela m'a permis de mener à bien les différentes étapes des tâches qui m'ont été confiées : effectuer un travail de recherche qui implique l'écriture d'un état de l'art, les implémentations des méthodes de l'état de l'art, les expérimentations et les discussions sur l'approche. Dans le cadre de mon stage de fin d'étude, ce rapport présente les travaux réalisés au cours de mon stage au sein du laboratoire LIRIS.

Ce stage s'inscrit dans un projet pluridisciplinaire entre les domaines de l'informatique et de la géographie. Dans ce travail, plusieurs disciplines sont mobilisées comme l'intelligence artificielle, l'informatique, la géomatique, la géographie et les humanités numériques. La problématique principale concerne l'extraction d'informations géographiques pour la représentation cartographique de documents textuels. Le but de ce stage est d'élaborer de nouvelles méthodes d'extraction d'informations spatiales et de modes adéquats de représentation et de mobilisation cartographiques.

L'objectif du projet dans lequel s'inscrit ce stage est de développer une méthode adaptée pour l'analyse d'informations géo-sémantiques (spatio-temporelles et thématiques) issues de romans ou de récits de voyages. Il s'agit de documents hétérogènes, de grandes tailles, qui posent plusieurs problèmes en termes de complexité sémantique et de combinatoire. Le projet cherche à extraire et à analyser des informations géographiques (noms de lieux, déplacements de personnages, relations entre différentes entités) afin de les analyser et de les cartographier. Un des défis de ce projet repose sur le fait que certains éléments du langage servant à exprimer ces informations peuvent souvent avoir plusieurs sens (ex : un verbe de déplacement peut être utilisé pour exprimer autre chose qu'un déplacement, etc.). De ce fait, l'hétérogénéité des documents et l'ambiguïté due au langage naturel est un verrou important pour les méthodes d'extraction automatique d'informations.

Afin de répondre à cette problématique, nous proposons de développer une méthode capable d'isoler les parties (ensemble de paragraphes ou de phrases) d'un texte ayant un intérêt pour y appliquer ensuite des méthodes d'extraction d'informations. Nous émettons une hypothèse selon laquelle l'application de ces méthodes sur des zones ciblées et réduites du texte doit permettre de réduire le nombre de faux positifs et ainsi améliorer les résultats.

Le rapport se structure de la façon suivante : la section 1 présente le laboratoire ainsi que l'équipe que j'ai intégré pour ce stage. La section 2 présente un état de l'art des méthodes basées sur les word embeddings pour la classification de phrases. La section 3 décrit la chaîne de traitement développée pour la construction d'un corpus d'apprentissage et l'entraînement de modèles pour la classification de phrases contenant

des informations géographiques. La section 4 présente les résultats de différentes expérimentations menées sur différents corpus et avec différentes méthodes. Enfin, la section 5 conclut ce rapport.

# 1. Présentation de l'organisme de stage

## 1.1. LabEx IMU

Ce stage est financé par le LabEx IMU[2] (Intelligences des Mondes Urbains) dans le cadre du pôle d'investigation « Matériaux, sources, données : savoirs et savoir-faire ». Ce Laboratoire d'Excellence est un dispositif de recherche et d'expérimentation centré sur les domaines de la ville, l'urbain, la métropolisation et l'urbanisation. Sa vocation est de stimuler, produire, capitaliser et valoriser une expertise scientifique et technique sur les mondes urbains passés, présents et à venir, tout en contribuant à l'action des pouvoirs publics et des acteurs privés. Tout comme le LIRIS, IMU associe un grand ensemble de domaines scientifiques. Il intègre à la recherche les praticiens des collectivités et institutions territoriales, des entreprises et des pôles de compétitivité. IMU a pour ambition de créer les conditions d'une intelligence collective dans les processus d'urbanisation et de métropolisation en assurant l'exercice de la pluralité scientifique radicale qui ouvre sur de nouveaux horizons de connaissances et d'innovations.

## 1.2. Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS)

Le laboratoire d'Informatique en Image et Systèmes d'Information (LIRIS) est une unité mixte de recherche en informatique avec comme 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 LIRIS est caractérisé par sa pluridisciplinarité et les recherches effectuées sont au service de problématiques sociétales importantes. Le laboratoire intervient à différents niveaux : local, régional et national, et entretient des rapports avec des entreprises dans divers projets collaboratifs. Depuis sa création, il joue un rôle important structurant l'informatique au niveau lyonnais. Le LIRIS est composé de six pôles de compétences qui participent à la valorisation des travaux de recherche. Ces six pôles de compétences regroupent 14 équipes (*Fig. 1.1*):

- Vision intelligente et reconnaissance visuelle (équipe Imagine)
- Géométrie et Modélisation (équipes M2DisCo et GeoMod)

[2] (n.d.). LabEx IMU. Retrieved August 15, 2020, from https://imu.universite-lyon.fr/

- Science des données (équipes BD, DM2L et GOAL)
- Interactions et cognition (équipes SMA,TWEAK et SICAL)
- Services, Systèmes distribués et Sécurité (équipes DRIM et SOC)
- Simulation, virtualité et sciences computationnelles (équipes Beagle, R3AM et SAARA)

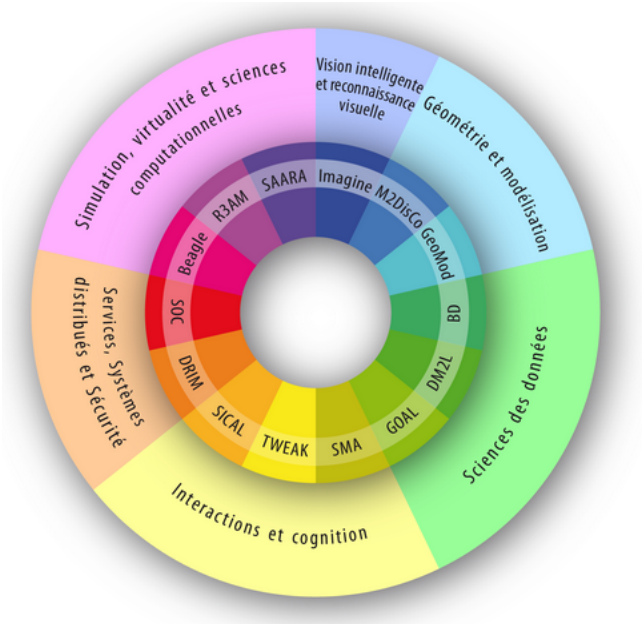



Fig. 1.1 : Les activités scientifiques de ses 14 équipes de recherche sont structurées en 6 pôles de compétences

Dans le cadre de ce stage, je travaille au sein de l'équipe DM2L (Data Mining et Machine Learning) qui étudie les principes fondamentaux des méthodes d'exploration de données complexes, telles que les données spatio-temporelles et les graphes dynamiques. L'équipe travaille au 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. Les sciences de la vie et la biologie moléculaire ont été particulièrement étudiés ces dernières années mais l'équipe développe également des travaux en collaboration avec des entreprises (thèses CIFRE ou FUI) ou d'autres partenaires académiques. Cette diversité est la conséquence d'une volonté de développer des algorithmes génériques applicables à un large spectre de données.

Mon encadrant, Monsieur Ludovic MONCLA, est maître de conférences en informatique à l'INSA Lyon. Ses thématiques de recherche 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. Ce stage s'inscrit également dans une collaboration avec Monsieur Thierry JOLIVEAU, professeur de géographie à l'université de Saint-Etienne.

Durant ce stage, du fait de la situation exceptionnelle liée au confinement et à la crise du COVID-19, nous avons travaillé à distance en organisant des réunions hebdomadaires

pour discuter des avancées et des orientations à suivre pour la semaine suivante. Nous avons partagé des documents et collaboré sur la labellisation des jeux de données grâce à Google Drive et nous avons également pu partager du code grâce à Google Colab et à un gitlab hébergé au LIRIS. De plus, chaque semaine pendant le confinement, une réunion d'équipe était organisée avec des présentations de travaux en cours des différents doctorants, post-doctorants ou stagiaires. Ces présentations étaient par exemple l'occasion pour les intervenants de s'entraîner pour une soutenance ou une conférence. J'ai pu notamment lors d'une de ces réunions présenter mon travail à l'ensemble de l'équipe de recherche. En plus de ces réunions, il y a eu deux réunions avec la direction du laboratoire ainsi que des réunions hebdomadaires avec les autres stagiaires du laboratoire. Cela m'a à nouveau permis de présenter mon travail aux autres stagiaires et de prendre connaissance de leurs travaux.

# 2. Etat de l'art

Une première partie de mon travail a consisté à identifier les travaux et méthodes existants en lien avec la problématique de segmentation automatique de texte. Le terme "segmentation de texte" peut avoir plusieurs sens. Il désigne le plus souvent le fait de segmenter une chaîne de caractères en phrases, en mots ou en unités lexicales en se basant notamment sur les espaces et la ponctuation. Dans le cadre de ce projet, on parle plutôt de segmentation thématique ou sémantique. La segmentation de texte signifie alors découper le texte en parties homogènes d'un point de vue thématique ou sémantique. Il s'agit donc de repérer des phrases ou ensembles de phrases traitant d'un sujet commun. Dans la littérature, on retrouve ce type d'approches pour des méthodes de détection de thématiques (topic modeling) (Jelodar et al., 2017), de résumé automatique. Cette tâche de segmentation automatique de texte est le plus souvent une étape de pré-traitement nécessaire pour des applications en traitement automatique du langage (TAL).

La sémantique ayant un rôle primordial, nous nous sommes intéressés aux méthodes récentes utilisant les *word embeddings* (Alemi & Ginsparg, 2015). Les *word embeddings* (ou *distributional semantic models*) permettent de représenter les mots par des vecteurs numériques (réelles) afin de les projeter dans un espace vectoriel. L'objectif de ces méthodes s'appuie sur leur capacité à capturer la sémantique de chaque mot en s'appuyant sur leur utilisation dans la langue à l'aide de corpus de textes (*Fig. 2.3*). Cela signifie que les variations de mots comme la casse ou l'orthographe sont automatiquement capturées lors de l'entraînement pour être similaires ou très proches dans l'espace des *embeddings*. Cela permet de simplifier les traitements (lemmatisation des mots) ou de traiter des corpus bruités (erreur OCR[3], fautes d'orthographe ou abréviations) (Baranes, 2012). L'utilisation de vecteurs de valeurs réelles pour représenter les mots d'un texte permet d'utiliser des méthodes d'apprentissage automatique et d'apprentissage profond adaptées. Comme illustré sur la (*Fig. 2.1*), les

[3] (n.d.). Optical character recognition - Wikipedia. Retrieved August 15, 2020, from https://en.wikipedia.org/wiki/Optical_character_recognition

*words embeddings* sont souvent utilisées comme une étape de prétraitement en amont d'une tâche de classification par exemple[4].

Les *word embeddings* sont des modèles pré-entraînés (*pre-trained embeddings*) par des réseaux de neurones à partir d'une très grande quantité de données (Qiu et al., 2020).

Il existe deux types principaux de *word embeddings* (*Fig. 2.2*) : non-contextuels (statiques) et contextuels (dynamiques). Dans le cas des *embeddings* statique, la première étape de la représentation du langage consiste à projeter les symboles du langage discrets dans un espace vectoriel. Chaque mot (ou sous-mot) d'un vocabulaire est associé à un vecteur.

Dans le type statique de *word embeddings*, (Mikolov et al., 2013) proposent Word2vec, un algorithme de *word embedding* utilisant deux architectures, CBOW (Continuous bag of words) et Skip-gram (*Fig. 2.4*).

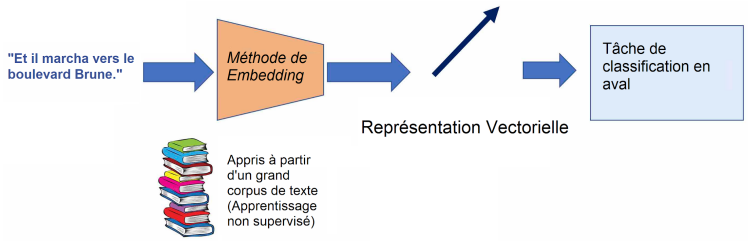



Fig. 2.1: Pipeline typique pour les méthodes non supervisées de *Word Embedding*

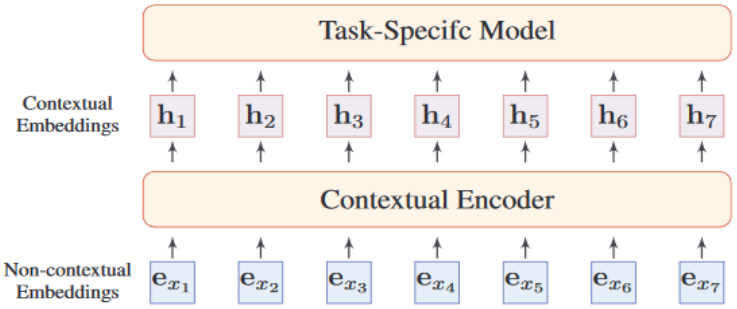
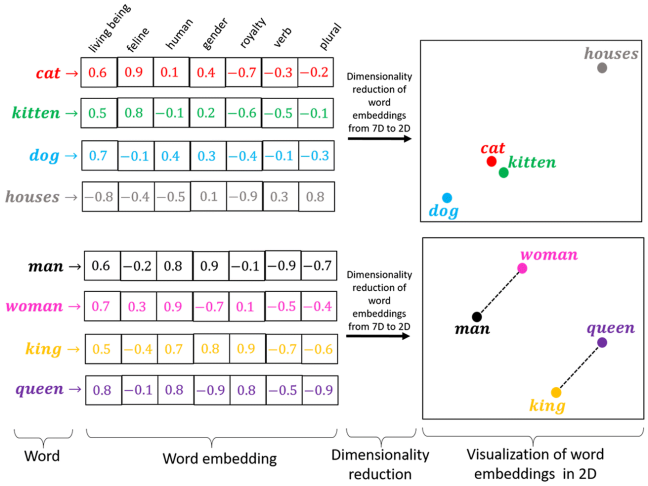



Fig. 2.2 : Architecture neuronale générique  pour le TAL (Qiu et al., 2020)

[4] (n.d.). Deep-learning-free Text and Sentence Embedding, Part 1 .... Retrieved August 15, 2020, from https://www.offconvex.org/2018/06/17/textembeddings/



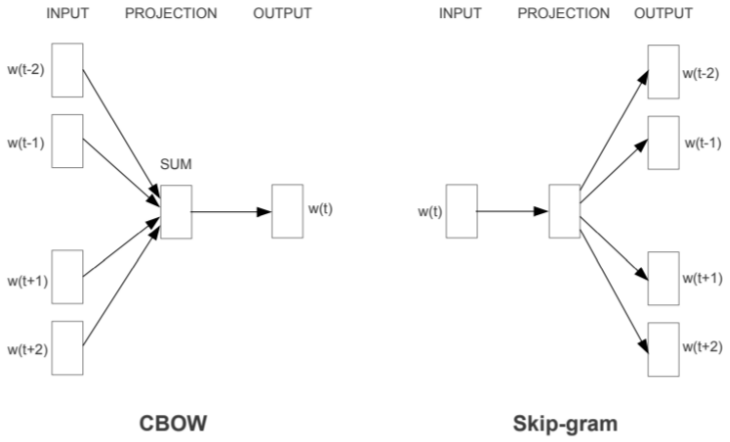




Fig. 2.3 : Illustration de la représentation des mots dans un espace vectoriel[5]

Fig. 2.4 : Représentation graphique des modèles CBOW et Skip-gram de Word2vec[6]

[5] (2020, February 13). Feature vectors for multiclass perceptron - Stack Overflow. Retrieved August 15, 2020, from https://stackoverflow.com/questions/60198443/feature-vectors-for-multiclass-perceptron

[6] (2017, September 20). (PDF) Apprentissage de représentations de documents et leur .... Retrieved August 15, 2020, from https://www.researchgate.net/publication/319930768_Apprentissage_de_representations_de_documents_et_leur_exploitation_en_recherche_d'information

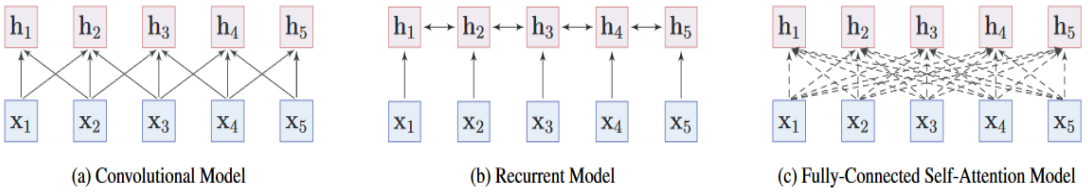



Fig. 2.5 : Encodeurs contextuels neuronaux (Qiu et al., 2020)

Si de nouvelles approches sont plus performantes, Word2vec permet de générer des *embeddings* de bonne qualité pour capturer les similarités syntaxiques et sémantiques latentes entre les mots. Parmi les dernières approches, GloVe (Pennington et al., 2014) est un modèle largement utilisé pour obtenir des modèles de langues pré-entraînés. Les *word embeddings* sont générés en s'appuyant sur des statistiques globales, plus particulièrement des co-occurrences de mot mesurées dans des grands corpus. Bien que ce type de modèles se soient révélés efficaces dans de nombreuses tâches de TAL, plusieurs limitations apparaissent. Ces algorithmes ne permettent pas de modéliser la polysémie et la prise en compte de différents contextes auxquels un même mot peut appartenir (*Fig.* 2.6). Par exemple, les différents sens du mot "cold" ("it's cold" vs "he is cold") sont combinés et réduits à un seul vecteur.Par ailleurs, lorsqu'ils sont utilisés sur une tâche spécifique, l'ensemble du modèle doit être pré-entraîné.

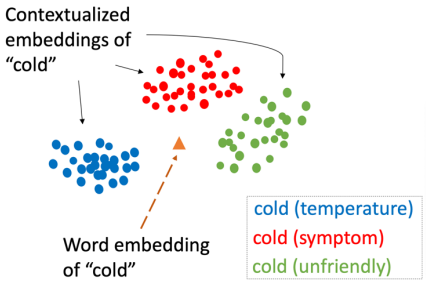



Fig. 2.6 : La prise en compte de différents contextes d'un mots dans le type dynamique de *word embedding* (Si et al., 2019)

Dans plusieurs chaînes de traitements automatiques des langues, l'unité d'analyse est la phrase, le paragraphe ou encore le texte dans sa globalité. Pour cela, de nombreux travaux tentent également d'entraîner des *embeddings* de paragraphes, de phrases ou de documents, comme le *paragraph vector* (Le & Mikolov, 2013), *Skip-thought vectors* (Kiros et al., 2015), et *Context2Vec* (Melamud et al., 2016). Une des principales différences de ces modèles de *sentence embeddings* est qu'ils tentent d'encoder des phrases dans un espace vectoriel de dimension fixe, plutôt que dans la représentation contextuelle de chaque token.

Pour répondre aux nombreuses limitations des modèles statiques, de nouveaux modèles de langue profonds contextuels ou modèles dynamiques sont proposés. Pour aborder la question des polysèmes et de la nature contextuelle des mots, nous devons distinguer la sémantique des mots dans différents contextes. Étant donné un texte où

chaque token est un mot ou un sous-mot, la représentation contextuelle du token dépend du texte entier.

L'idée générale de ce type de *word embeddings* est:

- Entraîner un modèle de langage profond
- Utiliser les représentations apprises par le modèle de langage dans les tâches de *downstream*, e.g. classification de texte.

La plupart des encodeurs de *Contextual Word Embeddings* peuvent être classés en deux catégories (Qiu et al., 2020): les modèles de séquence (*sequence models*) et les modèles basés sur des graphes (*graph-based models*).

La première catégorie contient des modèles convolutifs (*Convolutional Models*) (Kim, 2014)) et des modèles récurrents (LSTMs (Hochreiter & Schmidhuber, 1997), ELMO (Peters et al., 2018) (*Fig. 2.5*). Les modèles de cette catégorie apprennent la représentation contextuelle du mot avec un biais de localité et capturent difficilement les interactions à long terme entre les mots. Néanmoins, les modèles de séquence sont généralement faciles à entraîner et donnent de bons résultats pour diverses tâches de TAL.

Les modèles de la deuxième catégorie apprennent la représentation contextuelle avec une structure de graphe ou d'arborescence prédéfinie entre les mots, comme la structure syntaxique ou la relation sémantique. Certains modèles populaires de cette catégorie incluent Récursive NN (Socher et al., 2013), TreeL-STM (Tai et al., 2015) et GCN (Kipf & Welling, 2017). Plus récemment,de nouvelles approches incluent le mécanisme d'attention dépendant du *Transformer* (Vaswani et al., 2017) qui est un *Fully-Connected Self-Attention Model* qui n'utilise aucunes connexions récurrentes et qui utilise plutôt l'attention sur la séquence. Le mécanisme d'attention fait partie d'une architecture neuronale qui permet de mettre en évidence les caractéristiques pertinentes des données d'entrée[7]. Contrairement aux *sequence models*, le *Transformer* peut modéliser directement la dépendance entre chaque paire de mots dans une séquence. Actuellement, grâce à la puissance de *Transformer*, celui-ci est devenu l'architecture utilisée avec les modèles pré-entraînés les plus puissants.

Les modèles contextuels de *word embeddings* sont basés sur des architectures d'apprentissage profond. Le premier modèle pré-entraîné proposé de ce type est le modèle de l'article (Dai & Le, 2015). Les auteurs de cet article utilisent deux approches, la première consiste à initialiser les LSTM avec un modèle de langage (LM) et le deuxième consiste à initialiser un auto-encodeur de séquence. Ils montrent que les deux approches peuvent être utilisées comme des modèles pré-entraînés pour effectuer des tâches supervisées comme la classification de texte. Les récents modèles pré-entraînés ont des architectures plus profondes (LSTMs) et plus puissantes (Transformer). Ils sont entraînés avec des corpus beaucoup plus grands et de nouvelles tâches de pré-traitement. (Peters et al., 2018) proposent le modèle de représentation ELMo

[7] (n.d.). Attention in Natural Language Processing - arXiv. Retrieved August 15, 2020, from https://arxiv.org/pdf/1902.02181

(*Embeddings from Language Models*). Les représentations de ce modèle sont calculées par 2 couches de LSTM. Ce modèle est composé d'un *forward LM* (lire la séquence de gauche à droite) et d'un *backward LM* (lire la séquence de droite à gauche) (*Fig. 3.7*). Plusieurs améliorations sur un grand nombre de tâches de TAL ont été apportées par les représentations de ce modèle.

Les modèles très profonds, comme OpenAI GPT (*Generative Pre-training*) (Radford et al., 2018) et BERT (*Bidirectional Encoder Representation from Transformer*) (Devlin et al., 2019) ont montré leur grande capacité à apprendre des représentations universelles du langage. Depuis ce type de modèle (particulièrement BERT), la méthode de *Fine Tuning* est devenue l'approche la plus puissante pour adapter les modèles pré-entraînés à des tâches spécifiques.

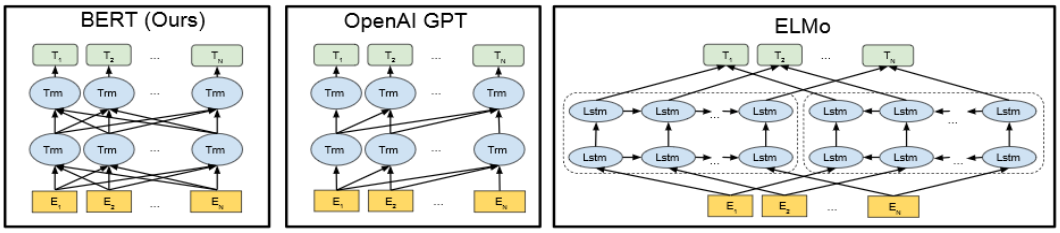



Fig. 2.7 : Différences dans les architectures des modèles pré-entraînés. (Devlin et al., 2019)

Comme le montre la (*Fig. 2.7*), BERT utilise un transformer bidirectionnel, OpenAI GPT utilise un *transformer* de gauche à droite et ELMo utilise la concaténation de LSTM de gauche à droite et de droite à gauche entraînés indépendamment pour générer des features pour les tâches en aval. Parmi les trois modèles, seules les représentations de BERT sont conditionnées conjointement sur le contexte gauche et droit dans toutes les couches. En plus des différences d'architecture, BERT et OpenAI GPT sont des approches de *fine-tuning*, tandis que ELMo est une approche *feature-based*. (Devlin et al., 2019)

Après avoir fait l'état de l'art sur les modèles existants de *word embeddings*, nous avons décidé de choisir un des modèles les plus récents qui prend surtout le contexte d'un mot en compte et qui utilise une architecture puissante (*Transformer*). Dans la prochaine section, nous allons voir en détails la sélection de notre modèle.

# 3. Processus de classification automatique de phrases

La problématique principale de ce travail est d'identifier de manière automatique dans un roman des zones de texte ayant une unité sémantique ou thématique spécifique. Plus particulièrement, on s'intéresse aux parties du texte qui comportent des informations géographiques liées au déplacement ou à la localisation des personnages. Pour satisfaire ce travail de segmentation thématique nous avons développé et évalué une méthode de classification automatique de phrases composée de deux étapes principales (*Fig. 3.1*).

La première a pour objectif la construction automatique d'un corpus labellisé. Ce corpus de phrases labellisées servira de jeu de données d'entraînement et d'évaluation pour la phase de classification supervisée. La deuxième étape de notre proposition concerne donc l'entraînement supervisé d'un modèle de classification. Notre objectif est d'identifier de manière automatique les phrases qui comportent des informations géographiques du type expression de déplacement associé à la présence d'un nom de lieu.

Notre méthode prend comme entrée des romans (grands textes) au format TXT ainsi que des fichiers XML (*Fig. 3.1*) contenant des annotations sémantiques réalisées de manière automatique par l'outil PERDIDO. Nous allons détailler ce corpus dans la section (4.1. résultats et expérimentations).

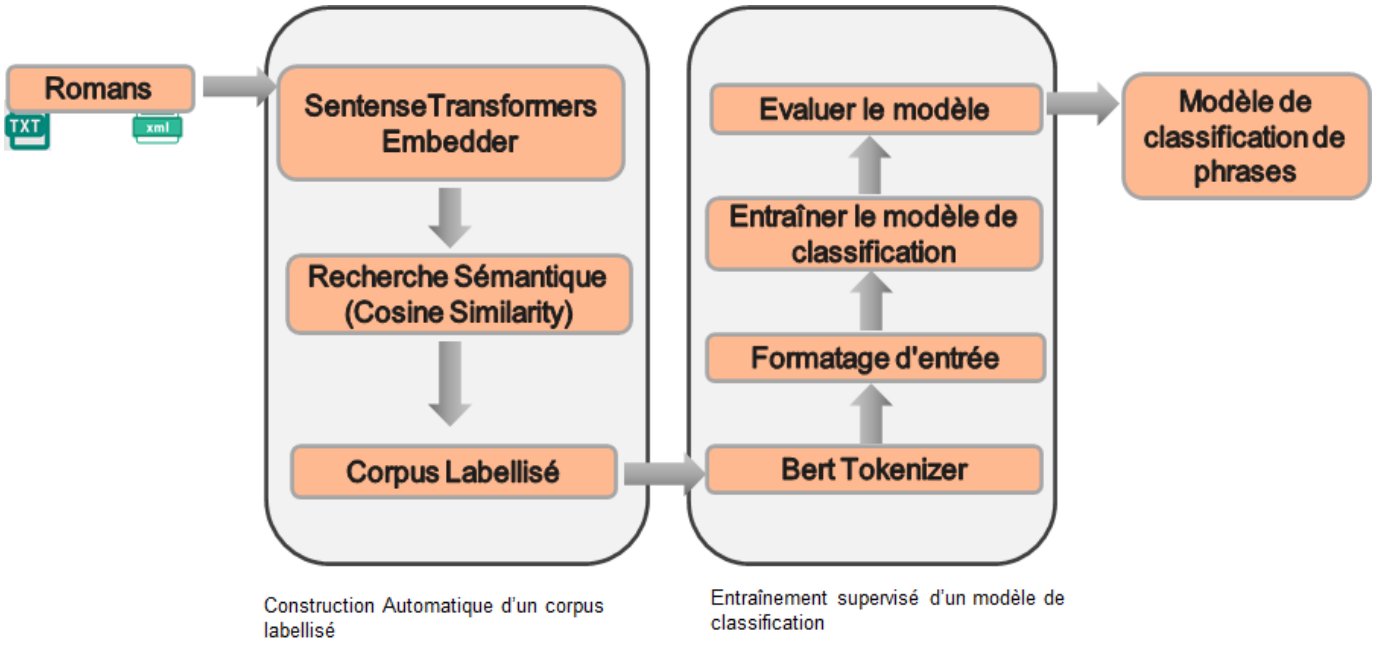



Fig. 3.1 : Schéma du processus de classification automatique de phrases

## 3.1. Construction automatique non-supervisée d'un corpus labellisé

Il s'agit de créer un jeu de données décomposé en deux parties : deux jeux de données, un pour l'entraînement et l'autre pour l'évaluation. Ce jeu de données contient deux colonnes : sentences, labels. Ces deux colonnes indiquent des phrases et des labels qui permettent de savoir si les phrases sont des indications géographiques ou non (label 0 : pas géographique, label 1 : géographique) (*Fig. 3.2*).

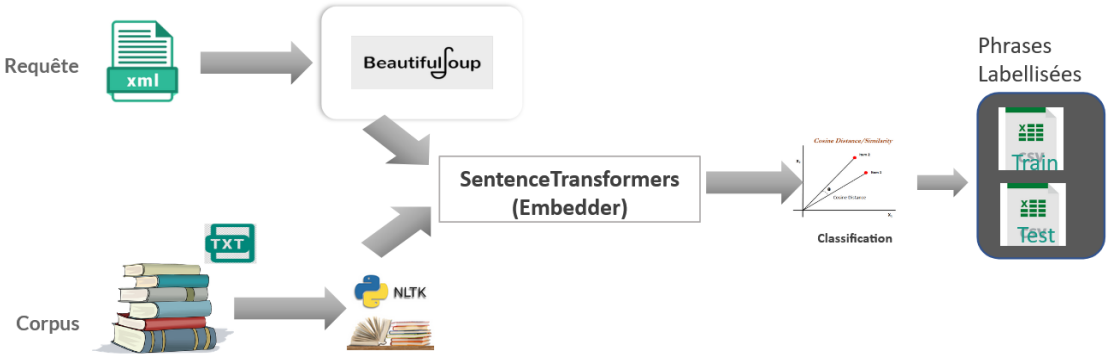



Fig. 3.2 : le schéma la construction automatique non-supervisée d'un corpus labellisé

### 3.1.1. Utilisation du géoparseur PERDIDO pour le repérage automatique de phrases géographiques

Dans ce travail nous avons souhaité pouvoir construire un corpus labellisé de manière automatique. Pour cela nous proposons dans un premier temps d'utiliser les annotations réalisées par le géoparseur PERDIDO[8]. PERDIDO est un outil d'annotation automatique qui utilise un système à base de règles expertes pour l'annotation d'informations géographiques (entités nommées, relations spatiales, expressions de déplacement). Il a été développé pour l'analyse de textes homogènes contenant exclusivement des informations géographiques. Notre objectif est de pouvoir l'appliquer aux romans en limitant le nombre de faux positifs dû à l'hétérogénéité et à la complexité des informations présentes. Nous faisons l'hypothèse que l'on peut utiliser les résultats même bruités (ou partiellement incorrects) pour construire un jeu de données d'entraînement pour la classification de phrases "géographiques". L'objectif est de segmenter le texte afin de faire un premier filtre et de repérer des parties du textes qui contiennent des informations géographiques. PERDIDO fait l'annotation des entités nommées, expressions de déplacements et relations spatiales et produit un résultat au format XML-TEI. La première

[8] (2017, March 20). Extended Named Entity Recognition Using Finite-State .... Retrieved August 15, 2020, from https://hal.archives-ouvertes.fr/hal-01492994/document

étape de notre traitement consiste donc à parser les fichiers XML produits pour chaque roman du corpus afin de repérer et d'extraire les phrases contenant des informations géographiques. Dans ce travail nous définissons une phrase "géographique" de manière simplifiée comme une phrase contenant un nom de lieu et un verbe de déplacement *(examples 1-3)*.

(1) "Vous allez au bois de Boulogne ou à Neuilly."
(2) "Et il marcha vers le boulevard Brune."
(3) "Ils traversèrent le pont d'Austerlitz, suivirent la ligne des boulevards, arrivèrent à la gare du Havre."

Afin d'extraire ce type de phrases à partir des XML (les annotations de PERDIDO), nous faisons une recherche des balises XML qui nous intéressent en utilisant la librairie Python *Beautiful Soup*[9]. Il s'agit d'une librairie permettant d'extraire des données de fichiers HTML et XML et de naviguer et modifier facilement un arbre. Les informations que nous recherchons sont annotées sémantiquement par les balises <rs type="place"> et <motion> respectivement pour les noms de lieux (entités spatiales) et les verbes de déplacement (*Fig 3.3*).

```
<s>
  <w lemma="on" type="PRO" xml:id="w5106">On</w>
  <motion type="final">
    <w lemma="venir" type="V" xml:id="w5107">venait</w>
  </motion>
  <w lemma="de" type="PREP" xml:id="w5108">d'</w>
  <w lemma=";" type="PUN" xml:id="w5114">;</w>
  <w lemma="le" type="DET" xml:id="w5115">la</w>
  <rs type="ene">
    <rs type="place" subtype="ene" start="25165" end="25185">
      <term type="place" start="25165" end="25168">
        <w lemma="rue" type="N" xml:id="w5116">rue</w>
      </term>
      <w lemma="du" type="PREPDET" xml:id="w5117">des</w>
      <rs type="unknown" start="25173" end="25185">
        <name type="unknown">
          <w lemma="poissonnier" type="N" xml:id="w5118">Poissonniers</w>
        </name>
      </rs>
    </rs>
  </rs>
```

Fig. 3.3: Balises <motion> et <rs type = "place"> pour une phrase géographique

Le repérage de ces éléments XML à partir de l'annotation géo-sémantique de PERDIDO nous aide à construire un ensemble de phrases constituant un premier corpus. Nous souhaitons enrichir cette liste de phrases afin de pouvoir améliorer les performances de PERDIDO et de ne pas uniquement se reposer sur ces annotations automatiques. Nous proposons, à partir de cette liste de phrases, de récupérer un autre ensemble de phrases issues des mêmes romans (*Tableau 2*) et ayant comme caractéristiques d'être proches "sémantiquement". Il peut s'agir de phrases ressemblantes ou similaires mais n'ayant pas été repérées grâce aux annotations de PERDIDO. L'objectif étant de construire un corpus d'apprentissage pour une classification supervisée, nous avons besoin de constituer un jeu de données contenant les deux classes de phrases à repérer : "géographique" et "non géographique". Pour cela, nous devons également ajouter des phrases "non géographiques" à notre corpus. Nous

[9] (2020, May 17). beautifulsoup4 · PyPI. Retrieved August 15, 2020, from https://pypi.org/project/beautifulsoup4/

détaillons dans la prochaine section la méthode implémentée pour la sélection automatique de phrases similaires et dissimilaires.

### 3.1.2. Sélection automatique de phrases similaires ou dissimilaires

Une fois un premier ensemble de phrases extrait du corpus de romans pour la classe “phrases géographiques”, nous souhaitons l’enrichir avec des phrases similaires non repérées par PERDIDO mais également sélectionner des phrases pour la classe “phrases non géographiques”.

Au cours de cette étape, nous avons tenté différentes méthodes (modèles) comme l’application de la recherche sémantique sur des *embeddings* des modèles non-contextual ou comme l’utilisation de *Multilingual Universal Sentence Encoder* (MUSE) (Yang et al., 2019) qui encode des phrases dans un espace vectoriel de dimension fixe. Mais nous avons finalement choisi d’appliquer *Sentence Transformers*[10] (qui permet de faire des *embeddings* de phrases (*embedding contextuel*)) sur les phrases des romans et les phrases d’entrées labellisées. Ensuite, nous appliquons la recherche sémantique (comparaison des vecteurs dans l’*embedding*) qui consiste à trouver des phrases similaires à une ou plusieurs phrases données. Nous effectuons la recherche sémantique en utilisant la similarité cosinus (*Fig. 3.4*).

$$cosine\ (S, Q)\ = \frac{S.Q}{||S||||Q||}$$

S: Sentence, Q: Query

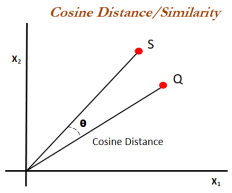



Fig. 3.4 : La similarité cosinus

Grâce aux vérifications manuelles effectuées, nous prenons pour chaque phrase donnée les 12 phrases les plus similaires et les 12 phrases les plus lointaines.

Pour trouver les phrases géographiques dans le corpus nous prenons les phrases qui sont répétées avec plus que 4 phrases données pour les phrases les plus similaires. Nous avons pris cette proposition après avoir effectué et documenté une vérification manuelle exhaustive des résultats en calculant la distance cosine pour réduire le nombre de faux positifs afin d’avoir un jeu de données bien labellisé pour l’étape suivante.

Pour les phrases non géographiques, après avoir obtenu les phrases géographiques, nous prenons environ (3 * nb de phrases géographiques) parmi les phrases du corpus des phrases les plus lointaines afin de préparer les jeux des données (celui de l’entraînement et celui de l’évaluation) avec la même structure.

[10] (n.d.). UKPLab/sentence-transformers: Sentence ... - GitHub. Retrieved August 15, 2020, from https://github.com/UKPLab/sentence-transformers

Nous utilisons le corpus (le jeu de l'entraînement et le jeu de l'évaluation) généré par cette méthode comme entrée de l'étape prochaine qui va entraîner un modèle de classification de phrases sur le jeu de l'entraînement et que nous allons l'évaluer sur le jeu de l'évaluation.

## 3.2. Classification automatique des phrases

Une fois le modèle entraîné et évalué sur le jeu d'évaluation, nous pouvons l'utiliser pour prédire des labels des phrases d'un roman (*Fig. 3.5*).

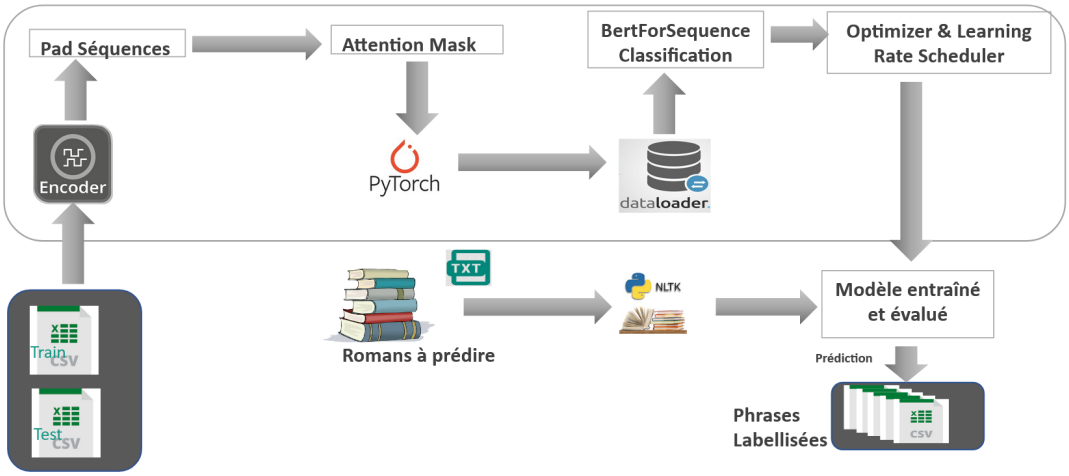



Fig. 3.5 : Schéma de la classification automatique des phrases

Après avoir vu les modèles dans les travaux connexes, nous avons décidé d'utiliser BERT qui est le modèle le plus puissant pour plusieurs tâches de *downstream* (notamment la classification de phrases (Sun et al., 2020) et qui a plusieurs modèles pré-entraînés sur différentes tâches et différentes langues.

Par la suite, nous allons voir dans la section (3.2.1.) les idées principales sur comment le modèle BERT a été créé, ensuite dans la section (3.2.2.), nous verrons la méthode de Fine Tuning sur les modèles de Bert en détails et ses avantages sur les tâches de TAL, de plus nous expliquerons l'utilisation de la Fine-Tuning pour la classification de phrases dans la section (3.2.3.), enfin, nous parlerons des modèles de Bert que nous avons utilisés dans la section (3.2.4.)

### 3.2.1. BERT (Bidirectional Encoder Representations from Transformers)

BERT est un modèle pré-entraîné pour des tâches de TAL développé par des chercheurs de Google en 2018. Ce qui le rend unique par rapport au reste des modèles, c'est qu'il est le premier modèle profondément bidirectionnel, que son type de représentation du langage est non supervisé et qu'il est pré-entraîné en utilisant

uniquement un corpus de texte brut. Avec BERT, nous n'avons pas besoins d'avoir des très grands corpus pour l'entraînement de notre modèle. Cela nous aide à économiser du temps, de l'énergie et des ressources.

BERT est pré-entraîné sur un grand corpus de texte non étiqueté qui comprend l'ensemble de Wikipédia (environ 2 500 millions de mots) et un corpus de livres (800 millions de mots) (Devlin et al., 2019). Les auteurs de BERT utilisent des *embeddings* de sous-mot (*WordPiece*) comme (Wu et al., 2016), ce qui aide à réduire la taille de vocabulaire qui est de 30 000 tokens et à augmenter la quantité de données disponibles. Pour la classification, ils utilisent le token spécial ([CLS]) qui est le premier token de chaque séquence. BERT s'appuie sur une architecture nommée Transformer. Contrairement aux approches précédentes (comme Elmo), cette architecture n'utilise pas du tout de connexions récurrentes mais plutôt l'attention sur la séquence. Pour chaque token, l'entrée de BERT est construite en additionnant les embeddings de tokens, les *embeddings* positionnelles, les *embeddings* de segments (*Fig. 3.6*). Sauf les *embeddings* de tokens, les autres types de *embeddings* sont des *embeddings* supplémentaires qui fournissent des métadonnées très importantes pour décrire les modèles pré-entraînés. les *embeddings* positionnelles expriment la position des mots dans une phrase, et les *embeddings* de segment aident le modèle à distinguer les phrases dans l'entrée (BERT peut être entraîné sur des paires de phrases pour les tâches qui prennent des paires de phrases en entrée).

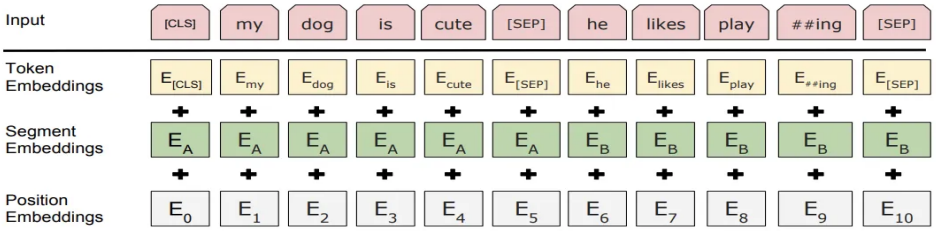



Fig. 3.6 : Schéma d'entrée pour BERT (Devlin et al., 2019)

Les auteurs de BERT utilisent deux stratégies pour créer leur modèle, la première est le modèle de langage masqué (*Masked Language Model* ou MLM) (*Fig. 3.7*) et la deuxième est la prévision de phrase suivante (*Next Sentence Prediction* ou NSP).


Fig. 3.7 : *Forward, Backward, and Masked Language Modeling*[11]

[11] (2019, May 11). Building a Multi-label Text Classifier using BERT and .... Retrieved August 15, 2020, from https://towardsdatascience.com/building-a-multi-label-text-classifier-using-bert-and-tensorflow-f188e0ecdc5d

### 3.2.2. BERT Fine-Tuning

La méthode *Fine-Tuning* signifie prendre les poids d'un réseau de neurones pré-entraîné et l'utiliser comme initialisation pour un nouveau modèle d'apprentissage sur nos données spécifiques. La *Fine-Tuning* sur Bert consiste généralement à ajouter une couche entièrement connectée au-dessus du BERT et à l'entraîner pendant quelques époques. Pour faire la *Fine-tuning* sur BERT, la plupart des hyperparamètres de modèles sont les mêmes, à l'exception de la taille du *batch* (*batch size*), de *learning rate* et du nombre d'époques de l'entraînement. Les valeurs optimales des hyperparamètres sont spécifiques à la tâche, mais les auteurs ont constaté que des valeurs peuvent fonctionner bien dans toutes les tâches (Devlin et al., 2019):

- *Batch size*: 16, 32
- *Learning rate* (Adam): 5e-5, 3e-5, 2e-5
- Nombre d'époques: 2, 3, 4

Nous pouvons obtenir plusieurs avantages avec Fine-Tuning[12]:

1. Développement plus rapide:

Avec la méthode de *Fine-tuning* sur BERT, nous avons besoin de beaucoup moins de temps pour entraîner un *fine-tuned* modèle grâce aux informations codées sur notre langue par les poids du modèle BERT. En fait, les auteurs recommandent seulement 2 à 4 époques de formation pour faire *fine-tuning* sur BERT pour une tâche spécifique de TAL.

2. Moins de données

Un ensemble de données beaucoup plus petit que ce qui serait requis dans un modèle construit à partir de zéro. Avec les modèles de TAL qui sont construits à partir de zéro et qui s'appuyant sur un réseau de neurones, nous avons souvent besoin d'un ensemble de données prohibitif afin d'entraîner notre réseau à une précision raisonnable, ce qui prend beaucoup de temps et d'énergie à la création de l'ensemble de données. Avec le *Fine-Tuning* de BERT, il est possible d'entraîner un modèle avec de bonnes performances sur une quantité raisonnable de données d'entraînement.

3. De meilleurs résultats

Avec *Fine-tuning* sur BERT, il a été démontré que cette méthode simple permet d'obtenir des résultats de l'état de l'art avec des ajustements spécifiques aux tâches minimales pour une grande variété de tâches de TAL (surtout classification). Plutôt que d'implémenter des architectures personnalisées et parfois obscures qui fonctionnent bien sur une tâche spécifique, simplement, *Fine-Tuning* sur BERT se révèle être une meilleure alternative (ou au moins égale).

[12] (2019, July 22). BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick. Retrieved August 15, 2020, from https://mccormickml.com/2019/07/22/BERT-fine-tuning/

La *Fine-Tuning* est généralement très rapide, il est donc raisonnable d'exécuter simplement une recherche exhaustive sur les paramètres ci-dessus et de choisir le modèle qui fonctionne le mieux sur l'ensemble de développement.

### 3.2.3. BERT Fine-Tuning pour la classification de phrases

L'implémentation *huggingface Pytorch* comprend un ensemble d'interfaces (classes) conçues pour une variété de tâches de TAL. Bien que ces interfaces soient toutes construites sur un modèle BERT pré-entraîné, chacune possède des couches supérieures et des types de sortie différents conçus pour s'adapter à une tâche spécifique de TAL.

La classe utilisée pour la classification de phrases s'appelle "BertForSequenceClassification". Il s'agit du modèle BERT normal avec une couche linéaire unique ajoutée pour la classification que nous utiliserons comme classificateur de phrases. Au fur et à mesure que nous alimentons les données d'entrée, nous entraînons l'ensemble du modèle BERT pré-entraîné et la couche de classification supplémentaire (non entraînée) sur notre tâche spécifique (*Fig. 3.8*).

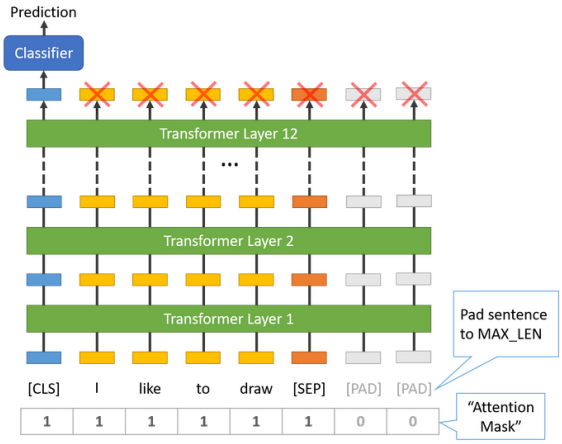



Fig. 3.8 : Structure de Bert avec les formatages requis pour l'entrée (Padding, Attention Mask, special tokens)

### 3.2.4. Sélection de modèles

Parmi la liste de modèles pré-entraînés de Bert (*Tableau 1*), nous utilisons des modèles de Bert qui sont pré-entraînés sur plusieurs langues (*Bert-multilingual*) qui sont les deux modèles (*bert-base-multilingual-cased, bert-base-multilingual-uncased*)[13].

| Architecture | Nom du raccourci | Détails du modèle |
|---|---|---|

[13] (n.d.). google-research/bert: TensorFlow code and pre ... - GitHub. Retrieved August 15, 2020, from https://github.com/google-research/bert

| | | |
|---|---|---|
| Bert | bert-base-uncased | 12-layer, 768-hidden, 12-heads, 110M parameters.<br>Trained on lower-cased English text. |
| | bert-large-uncased | 24-layer, 1024-hidden, 16-heads, 340M parameters.<br>Trained on lower-cased English text. |
| | bert-base-cased | 12-layer, 768-hidden, 12-heads, 110M parameters.<br>Trained on cased English text. |
| | bert-large-cased | 24-layer, 1024-hidden, 16-heads, 340M parameters.<br>Trained on cased English text. |
| | bert-base-multilingual-uncased | (Original, not recommended) 12-layer, 768-hidden, 12-heads, 110M parameters.<br>Trained on lower-cased text in the top 102 languages with the largest Wikipedias |
| | bert-base-multilingual-cased | (New, recommended) 12-layer, 768-hidden, 12-heads, 110M parameters.<br>Trained on cased text in the top 104 languages with the largest Wikipedias. |

Tableau 1: Modèles pré-entraînés les plus connus du Bert dans Huggingface[14]

Avant d'appliquer la classe *BertForSequenceClassification* pour la classification de phrases, nous passons par une étape de formatage d'entrée (*Fig 3.8*). Dans cette étape, nous ajoutons les formatages requis pour l'entrée des modèles de Bert comme ajouter les tokens spéciaux, faire la Pad-séquence sur les phrases d'entrée afin d'appliquer le modèle sur une taille fixe de vecteurs de représentations de phrases, et enfin différencier explicitement les vrais tokens de tokens après l'application de Pad-séquence avec le masque d'attention (*attention mask*).

Après avoir appliqué les formatages requis, nous transformons les données aux données de type Pytorch pour pouvoir appliquer la classe *BertForSequenceClassification.* Enfin, nous saisissons les hyperparamètres de l'entraînement à partir du modèle stocké (*Fig. 3.5*).

## 3.3. Techniques et outils utilisés

Afin de réaliser au mieux les deux phases décrites précédemment, des outils techniques sont mobilisés pour atteindre les objectifs du stage. Cette partie a pour but de

[14] (n.d.). Pretrained models - Hugging Face. Retrieved August 15, 2020, from https://huggingface.co/transformers/pretrained_models.html

décrire les différentes techniques et outils utilisés pendant ce stage ainsi qu'une explication de l'intérêt de chacun. Nous avons choisi de classer les outils selon des caractéristiques communes.

La première classe correspond au langage de programmation et de Sciences des Données Python et ses bibliothèques. Parmi les bibliothèques de Python, nous avons utilisé les suivantes:

- Pandas : Pandas est un outil d'analyse et de manipulation de données open source rapide, puissant, flexible et facile à utiliser, construit au-dessus du langage de programmation Python.
- Scipy spatial : SciPy est un logiciel open source pour les mathématiques, les sciences et l'ingénierie. Je l'utilise dans l'étape « non supervisée » pour calculer la similarité entre les phrases issues du parsing XML et les phrases de romans.
- Beautifulsoup : Elle est une bibliothèque Python pour extraire des données de fichiers HTML et XML. Je l'utilise lors de la première tâche pour interroger les fichiers XML annotés.
- Scikit-learn: Il s'agit d'une bibliothèque d'apprentissage automatique de logiciels gratuits pour le langage de programmation Python. Je l'utilise pour la classification (split les données, utiliser les modèles classiques de classification comme SVM et Naïve-bayes, utiliser les métriques de performance) ainsi que les libraries Numpy, Matplotlib, etc.
- NLTK : Natural Language Toolkit (NLTK) est une bibliothèque en Python permettant un traitement automatique des langues. Je l'utilise avec le module "re" (Regular Expression) pour tokenizer les phrases des romans.
- Keras : Elle est une bibliothèque open source réseau-neuronal conçue pour permettre une expérimentation rapide avec les réseaux de neurones profonds. Je l'utilise pour faire le *padding* sur les séquences des *embedding*.
- Tensorflow : Elle représente une bibliothèque mathématique symbolique, également utilisée pour les applications d'apprentissage automatique telles que les réseaux de neurones. Son architecture flexible permet le déploiement facile du calcul sur une variété de plates-formes (CPU, GPU, TPU). Je l'utilise dans l'étape de classification pour utiliser la plateforme GPU sur Google Colab.
- PyTorch : Pytorch est une bibliothèque d'apprentissage automatique open source basée sur la bibliothèque Torch, utilisée pour des applications telles que la vision par ordinateur et le traitement du langage naturel. Je l'utilise dans l'étape de classification de texte (supervisée) parce qu'elle a de bons wrappers et des modèles pré-entraînés de BERT.

La deuxième classe correspond à l'environnement Colab et son cloud. Colab ou Colaboratory est un environnement de notebook Jupyter gratuit qui ne nécessite aucune configuration et fonctionne entièrement dans le cloud. Avec Colab, on peut importer un jeu de données de texte, y former un classifieur de texte et évaluer le modèle, le tout en quelques lignes de code. Colab exécute du code sur les serveurs cloud de Google, ce qui signifie qu'il est possible de tirer parti de la puissance du matériel Google, y compris les GPU et TPU, quelle que soit la puissance de notre machine. Tout ce dont nous avons besoin est un navigateur.

La troisième classe correspond à *Sentence Transformers*[15] (utilisé pour la première étape) et permet d'effectuer *Fine-Tuning* sur BERT avec une structure de réseau siamois ou triplet pour produire des incorporations de phrases sémantiquement significatives qui peuvent être utilisées dans des scénarios non supervisés: similarité textuelle sémantique via *cosine-similarity*.

La dernière classe correspond à la Librairie *huggingface* (*Transformers*). L'implémentation *huggingface Pytorch* comprend un ensemble d'interfaces (classes) conçues pour une variété de tâches de TAL. Bien que ces interfaces soient toutes construites sur un modèle BERT pré-entraîné, chacune a des couches supérieures et des types de sortie différents conçus pour s'adapter à une tâche spécifique de TAL.

# 4. Expérimentations et Résultats

Toutes les expérimentations effectuées sont dirigées pour valider l'efficacité de notre modèle pré-entraîné avec BERT. Celui-ci est entraîné sur un petit corpus (quatre romans) du même écrivain. De ce fait, les quatre romans ont le même style d'écriture et des types de phrases proches. Lors des expérimentations nous avons entraîné un modèle sur un corpus plus grand et plus diversifié (auteurs, longueurs, années de publication, etc) afin de comparer et nous avons également entrainés des modèles à partir d'approches d'apprentissage automatique classiques.

## 4.1. Corpus / Données

Les expérimentations sont menées sur un corpus de romans français du XIXème siècle. Il s'agit de documents hétérogènes, de grande taille.

### 4.1.1. Le corpus de l'étape Construction d'un corpus labellisé

Nous possédons deux corpus pour cette étape. Le premier corpus est constitué de 4 romans et le deuxième contient 10 romans (*Tableau 2*), afin de créer deux jeux pour l'entraînement (le corpus d'apprentissage de l'étape de classification). Le modèle entraîné sur un grand corpus a pour but de vérifier l'efficacité du modèle entraîné sur un petit corpus. Si les résultats des deux modèles sont assez proches, alors le modèle est efficace sur le petit corpus.

| de roman | Auteur | Date de publication | Nb de phrases | Corpus |
|---|---|---|---|---|
| L'Assommoir | Emile Zola | 1877 | 10203 | **PCEN**, GCEN |

[15] (n.d.). UKPLab/sentence-transformers: Sentence ... - GitHub. Retrieved August 15, 2020, from https://github.com/UKPLab/sentence-transformers

| | | | | |
|---|---|---|---|---|
| Nana | Emile Zola | 1880 | 10292 | **PCEN**, GCEN |
| Au bonheur des dames | Emile Zola | 1883 | 8871 | **PCEN**, GCEN |
| L'Oeuvre | Emile Zola | 1886 | 9777 | **PCEN**, GCEN |
| Jack | Alphonse Daudet | 1876 | 8812 | PCEV, GCEN |
| Sans cravate | Paul de Kock | 1844 | 22874 | PCEV, GCEN |
| Pot-Bouille | Emile Zola | 1882 | 10207 | PCEV, GCEN |
| Bel-Ami | Guy de Maupassant | 1885 | 7083 | GCEN |
| L'Education sentimentale | Gustave Flaubert | 1869 | 10687 | GCEN |
| Les demoiselles de Magasin | Paul de Kock | 1863 | 13209 | Nom GCEN |
| Monsieur Bergeret à Paris | Anatole France | 1901 | 4491 | GCEV |
| La Charpente | J.H. Rosny Jeune | 1900 | 5241 | GCEV |
| Monsieur Choublanc à la recherche de sa femme | Paul de Kock | 1856 | 10909 | GCEV |
| Le père Goriot | Honoré de Balzac | 1835 | 6351 | GCEV |
| Histoire de la grandeur et de la décadence de César Birotteau | Honoré de Balzac | 1837 | 5993 | GCEV |
| Dans les rues | J.H. Rosny Aîné | 1928 | 7458 | GCEV |
| Sapho : Moeurs parisiennes | Alphonse Daudet | 1884 | 2117 | GCEV |
| Le ventre de Paris | Albert Robida | 1873 | 7782 | GCEV |
| Le Vingtième siècle - la vie électrique | Albert Robida | 1890 | 4000 | GCEV |
| Le vingtième siècle | Albert Robida | 1883 | 6914 | GCEV |

Tableau 2: **PCEN** (Petit Corpus d'Entraînement) : 4 romans de l'entraînement du même écrivain, GCEN (Grand Corpus d'Entraînement) : 10 romans du grand corpus de

plusieurs écrivains, PCEV (Petit Corpus d'Evaluation): 3 romans, GCEV (Grand Corpus d'Evaluation): 10 romans de grand corpus de l'évaluation

Nous possédons également deux corpus constitués respectivement de 3 romans et de 10 romans (*Tableau 2*), afin de créer deux jeux pour l'évaluation. Nous avons fait deux jeux de l'évaluation afin de voir si nous évaluons notre modèle sur des corpus ayant plusieurs types d'écriture de plusieurs écrivains.

Pour chacun de ces corpus, les requêtes utilisées dans cette étape sont les phrases qui contiennent des informations géographiques et qui sont repérées par Perdido à partir des mêmes corpus.

### 4.1.2. Le corpus de la classification de phrases

Nous effectuons nos expérimentations sur les corpus produits à la sortie de l'étape précédente qui contiennent tous les mêmes colonnes (sentences, labels). Comme nous l'avons indiqué précédemment, ces deux colonnes indiquent des phrases et des labels qui disent si les phrases sont des indications géographiques ou non (label 0 : pas géographique, label 1 : géographique).

Les 4 corpus possèdent la même structure (la même proportionnalité entre les phrases géographiques et non géographiques). Pour chacun des corpus, les phrases géographiques ~ ⅓ de phrases non géographiques (*Fig. 4.1*).

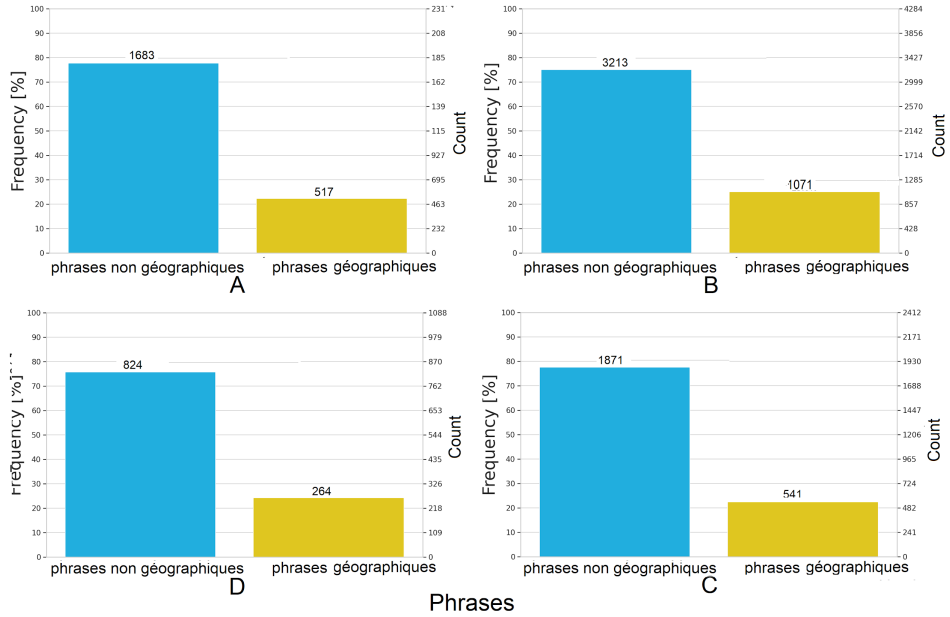



Fig. 4.1:
Visualisations des 4 corpus. A et B, C, D sont des échantillons qui viennent respectivement de PCEN, de GCEN, de PCEV et de GCEV (*Tableau 2*).

## 4.2. Construction automatique d'un corpus labellisé

Comme nous créons les deux corpus automatiquement, nous obtenons quelques phrases qui ne sont pas correctement labellisées (exemples 4-6). Afin d'avoir un corpus d'évaluation (pour la prochaine étape) entièrement correct, nous effectuons une vérification manuelle sur le corpus du test.

(4) "Le lendemain, neuf heures venaient de sonner, lorsque Tobie se présente à la porte d'ALbert et demande au domestique si son ami est visible."
(5) "Sanscravate est en un moment dans la rue, puis il court sans s'arrêter jusques au logement où il a laissé sa soeur."
(6) "Il arrive devant la porte de madame Baldimer, il sonne, la domestique paraît."

Nos phrases (géo, non géo) sont choisies parmi les phrases repérées en appliquant la recherche sémantique entre les *embeddings* du corpus et les *embeddings* de la requête. Cette recherche utilise la fonction *cosine distance* pour trouver la similarité entre les vecteurs de représentation des phrases (c'est-à-dire entre les phrases). Pour les phrases geo, comme nous l'avons indiqué précédemment, nous prenons les phrases répétées comme similaires avec plus que 4 phrases, et pour les phrases non géo, parmi les phrases les plus dissimilaires (sans duplications), nous prenons ~(3 * nb(géo)) phrases non géo.

Pour la phrase :
*"Celui -ci, pour se rapprocher de sa maîtresse, était venu demeurer rue Saint-Roch."*
les 3 phrases les plus similaires sont :

- *"Il traversa la Seine, il suivit toute la rue Saint-Jacques."*
- *"Et il raconta comment ils s'étaient rencontrés rue Rochechouart."*
- *"Rue de la Michodière, il allait rentrer chez lui, lorsqu'un boutiquier voisin, debout sur la porte, l'appela d'un signe."*

La phrase :
*"Alors, Satin voulut lui montrer sa porte ; elle demeurait à côté, rue La Rochefoucauld."*
apparaît comme similaire avec 4 phrases issues du XML :

- *"Nous en avons pour deux minutes, expliqua Baudu, pendant qu'il descendait la rue Gaillon avec sa nièce."*
- *"Heureusement, l'omnibus du boulevard Rochechouart à la Glacière passait près de l'asile."*
- *"Quatre hommes de bonne volonté transportèrent le blessé chez un pharmacien de la rue Gaillon, pendant que l'omnibus reprenait lentement sa marche."*
- *"Sur le boulevard, Gervaise tourna à gauche et suivit la rue Neuve de la Goutte-d'Or."*

Afin d'évaluer l'efficacité de la première étape de notre méthode, nous créons la requête à partir des XML des mêmes romans que ceux du corpus (TXT). Les phrases de la requête sont des phrases géo trouvées par PERDIDO. Nous effectuons une expérimentation sur les deux petits corpus (PCEN et PCEV *Tableau 2*). Pour le corpus d'apprentissage (train), nous avons obtenu les résultats suivants : un échantillon de 2200 phrases extraites de 4 romans qui contient 517 phrases géo, dont 403 phrases issues

directement des XML PERDIDO et 114 phrases récupérées en plus par la méthode, et enfin 1683 phrases “non géo”.

Pour le corpus de l’évaluation: nous avons obtenu les résultats suivants: un échantillon de 1088 phrases extraites de 3 romans qui contient 289 phrases géo, dont 251 phrases issues directement des XML PERDIDO, 38 phrases récupérées en plus par la méthode, une correction manuelle pour l’évaluation afin d’être sûr que les phrases sont correctement labellisées. Nous possédons 264 phrases géo après vérification, et enfin 824 phrases non géo.

Nous utiliserons ces deux corpus pour la prochaine étape en tant que train et test *datasets*.

## 4.3. Classification automatique des phrases

Nous effectuons nos expérimentations sur les deux corpus de l’étape précédente qui possèdent tout deux les mêmes colonnes (sentences, labels). Ces deux colonnes indiquent des phrases et des labels qui permettent de savoir si les phrases sont des indications géographiques ou non (label 0 : non géographique, label 1 : géographique).

Comme nous l’avons indiqué précédemment, nous utilisons deux modèles multilingues de BERT (*cased*, *uncased*) avec lesquels nous effectuons nos expérimentations en appliquant la méthode de BERT *Fine Tuning* avec Pytorch.

### 4.3.1. BERT - Hyperparamètres

Comme nous l’avons indiqué précédemment, nous utilisons les modèles de *Bert-base-multilingual* avec une taille cachée (*hidden size*) de 768, ainsi que 12 blocs *Transformer* et 12 têtes de *self-attention*. Nous entraînons nos modèles avec une *batch size* de 32, *max sequence length* de 256, un *learning rate* de 2e-5. Nous fixons empiriquement le nombre maximum de l'époque à 4 et enfin nous utilisons Adam avec 1e-8.

### 4.3.2. Métriques de performance

Afin d'évaluer les modèles, nous utilisons quatre métriques différentes. L'exactitude, Eq. (1), peut être considérée comme le taux de réussite global de la méthode en termes de prédiction. La précision, Eq. (2), est le rapport des prédictions positives correctement classées. le rappel, Eq. (3), est le rapport des cas réellement positifs qui sont également identifiés comme tels. Et enfin F1-score, Eq. (4), correspond à la moyenne harmonique de la précision et du rappel et donne une meilleure mesure des observations incorrectement classées.

$$\text{Exactitude} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$\text{Précision} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{Rappel} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F1-Score} = 2 \times \frac{Rappel \times Précision}{Rappel + Précision} \quad (4)$$

- TP est le nombre de vrais positifs (Phrases géographiques classées comme telles),
- TN le nombre de vrais négatifs (Phrases non géographiques classées comme telles),
- FP le nombre de faux positifs (Phrases non géographiques classées comme phrases géographiques),
- FN le nombre de faux négatifs (Phrases géographiques classées comme phrases non géographiques).

### 4.3.3. Processus automatique vs Processus semi-automatique

Nous faisons des expérimentations sur les deux processus (automatique et semi-automatique). Les deux processus sont assez proches selon plusieurs métriques d'évaluation (*Tableau 3*). Cela montre que nous pouvons toujours prendre le processus automatique pour entraîner nos modèles.

| Processus | Modèle | Accuracy | Precision | Recall | f1-score | Tp,Tn, Fp, Fn |
|---|---|---|---|---|---|---|
| Semi-Automatique | Bert cased | 99.72 | 100.00 | 98.62 | 99.25 | 264, 821, 0, 3 |
| | Bert uncased | 98.16 | 96.90 | 94.60 | 95.49 | 256, 812, 8, 12 |
| Automatique | Bert cased | 97.79 | 100.00 | 90.59 | 94.79 | 264, 800, 0, 24 |
| | Bert uncased | 99.72 | 100.00 | 98.62 | 99.25 | 264, 821, 0, 3 |

Tableau 3 : Performance des deux processus automatique et semi-automatique sur le corpus de 4 romans pour les modèles entraînés sur le petit corpus (4 romans)

### 4.3.4. Modèle petit corpus vs Modèle grand corpus, Modèles classiques

Afin de valider l'efficacité du modèle entraîné sur le petit corpus (4 romans), nous avons également entraîné un autre modèle sur un corpus plus grand (10 romans) (*Tableau 2*). Dans un premier temps, nous comparons l'évaluation des deux modèles Bert (*cased* et *uncased*) sur le grand corpus de l'évaluation (qui contient différents styles d'écriture de phrases d'auteurs différents) (*Tableau 4*). La différence entre les deux modèles n'est pas significative en termes d'écart d'accuracy ou de f1-score. Les résultats sont très bons (accuracy autour de 97%) et la modèle Bert *cased* obtient une précision de 100%. Ces résultats sont très proches de ceux obtenus sur le corpus de 4 romans (du même auteur). Cela montre que l'approche de l'utilisation d'un corpus de taille limité

permet déjà d'obtenir des bons résultats et qu'il n'est pas nécessaire d'avoir un très grand corpus.

Afin de vérifier l'efficacité des modèles de *Word Embeddings* que nous avons utilisé, nous implémentons plusieurs modèles classiques de l'apprentissage automatique comme SVM, *Random Forest* et Naïve-Bayes sur notre jeu de données en utilisant les représentations de TF-IDF. Nous utilisons un article pour la classification de texte avec Scikit-Learn[16] pour préparer les données et les représenter avec TF-IDF, et pour les hyperparamètres du modèle *Random Forest*. Concernant SVM, nous utilisons le modèle linéaire. Les modèles Bert donnent des résultats meilleurs que les modèles classiques.

| La taille de corpus de l'entraînement | Modèle | Accuracy | Precision | Recall | f1-score | Tp,Tn, Fp, Fn |
|---|---|---|---|---|---|---|
| 10 Romans (différents auteurs) | Bert cased | 97.16 | 100.00 | 89.25 | 94.00 | 541, 1802, 0, 69 |
| | Bert uncased | 96.95 | 99.67 | 88.88 | 93.61 | 539, 1799, 2, 72 |
| 4 Romans (de même auteur) | Bert cased | 96.95 | 100.00 | 88.71 | 93.63 | 541, 1797, 0, 74 |
| | Bert uncased | 97.03 | 99.29 | 89.10 | 93.54 | 539, 1801, 2, 70 |
| | Naive_bayes | 93.20 | 95.39 | 74.19 | 83.46 | 414, 1834, 20, 44 |
| | Random Forest Classifier | 96.10 | 96.58 | 86.20 | 91.09 | 481, 1837, 17, 77 |
| | SVC_Linear | 96.35 | 96.81 | 87.09 | 91.69 | 486, 1838, 16, 72 |

Tableau 4 : Comparaison de performances sur le grand corpus de l'évaluation (10 romans) des modèles entraînés sur le petit corpus avec les modèles entraînés sur le grand corpus. Seconde comparaison avec les modèles classiques de l'apprentissage automatique qui sont entraînés sur des représentations de TF-IDF de notre petit corpus.

Comme nous pouvons le constater dans le (*Tableau 4*) les modèles entraînés sur le petit corpus, construit de 4 romans ayant le même auteur (Zola) (PCEN *Tableau 2*), ont les mêmes performances que les modèles entraînés sur le grand corpus. Cela montre que Bert fonctionne efficacement même s'il n'est entraîné qu'avec un corpus d'un seul auteur (qui a le même type d'écriture). Nous pouvons ainsi supposer qu'il n'est pas

[16] (n.d.). Text Classification with Python and Scikit-Learn - Stack Abuse. Retrieved August 15, 2020, from https://stackabuse.com/text-classification-with-python-and-scikit-learn/

nécessaire d'entraîner les modèles sur un grand corpus si on utilise BERT pour la classification de phrases géographiques.

## 4.4. Labelliser les phrases d'un roman complet

Une dernière expérimentation réalisée correspond à la labellisation de toutes les phrases du roman Maison et à la vérification de ces résultats manuellement afin de vérifier la performance de nos modèles à bien prédire les labels. Ce roman est écrit par l'auteur Honoré de Balzac, il a été publié en 1830 et comprend 1010 phrases. Après avoir appliqué le modèle qui utilise *Bert-multilingual-uncased* (entrainé sur le petit corpus de 4 romans) sur ce roman, nous avons trouvé 42 phrases labellisées comme des phrases géographiques et 977 phrases labellisées comme des phrases non-géographiques. Après la vérification manuelle, nous avons trouvé:

- 13 phrases sur les 42 phrases géographiques sont non-géographique (FP), environs 30% de FPs.
- 5 phrases sur les 977 phrases non-géographique sont géographique (FN)

Après les bons résultats que nous avons obtenu en évaluant notre modèle sur les corpus d'évaluation construits par la méthode de première étape, la raison pour laquelle le pourcentage est de 30% de faux positifs réside dans le fait que le nombre de phrases géographique dans le roman Maison est beaucoup plus petit que le nombre de phrases non géographiques. Tandis que les corpus construits par la première étape ont la structure de (nombre de phrases géographiques ~ ⅓ de nombre de phrases non géographique).

Cela montre que notre modèle donne très bons résultats sur des corpus ayants la même structure que notre modèle.

## 4.5. Appliquer Perdido

Afin de vérifier l'efficacité de cette étape (ce stage) du projet et de répondre à la problématique, nous appliquons Perdido (l'outil de geoparsing) sur les phrases du roman Maison et nous créons à partir de cette application un jeu de données qui contient 3 colonnes, la première contient les phrases, la deuxième contient les labels et la troisième renseigne les résultats de Perdido (les entités nommées de lieux). Après la création de ce jeu de données, nous effectuons une vérification manuelle afin de voir si l'application de Perdido sur les parties géographiques du texte réduit le nombre de Faux Positifs. Avec les phrases non géographiques, Perdido trouve des entités de non-lieux en tant que des entités de lieux. 94 entités fausses ont été détectées comme des entités de lieux dans les 977 phrases négatives (85 Augustine) (~ 9%). Cela montre que si nous appliquons Perdido uniquement sur les phrases géographiques trouvées par la méthode, nous réduisons le nombre de faux positifs.

# 5. Conclusion et perspectives

L'objectif général de ce stage était de développer une méthode adaptée pour l'extraction d'informations géo-sémantiques issues de documents textuels (romans). Nous avions comme problématique principale l'hétérogénéité de ces documents et l'ambiguïté de la langue qui est un verrou important pour les méthodes d'extraction automatique d'informations. Afin d'appliquer ces méthodes sur des données homogènes et génériques, l'hypothèse était de segmenter le texte afin de faire un premier filtre et de repérer les parties du texte qui contiennent des informations géographiques. Au cours de ce stage, nous avons réalisé deux grandes phases. La première consistait à élaborer et à rédiger un état de l'art sur des méthodes (modèles) de *word embeddings* et d'apprentissage profond pour la segmentation automatique de texte ou la classification automatique de texte. La deuxième phase de ce stage était de choisir et d'adapter un de ces modèles. Parmi ces modèles, nous avons choisi le modèle BERT pour construire notre méthode. Le modèle BERT est un modèle pré-entraîné de *word embeddings* contextuels qui s'appuie sur l'architecture puissante *Transformer*. Nous avons utilisé la méthode efficace *Fine-Tuning* sur BERT car elle possède plusieurs avantages, par exemple, elle est rapide pour créer et entraîner le modèle et elle a besoin d'une quantité raisonnable de données pour offrir de bonnes performances. En évaluant un modèle de BERT qui est entraîné sur un petit corpus, nous possédons des résultats (100% de précision) plus performants que les résultats de modèles classiques (96.8% de précision avec SVM) d'apprentissage automatique de classification de texte. Pour répondre à la problématique, nous avons appliqué une des méthodes d'extraction automatique d'informations (l'outil de geoparsing Perdido) sur nos résultats et nous avons observé que l'application de Perdido uniquement sur les parties géographiques de texte réduit le nombre de faux positifs. L'ensemble du code développé pour les expérimentations est disponible en ligne sur le serveur gitlab du projet[17].

Nous avons également remarqué que notre modèle donne une très bonne performance uniquement lorsqu'il est appliqué sur des corpus de données ayants la même structure que notre modèle. C'est-à-dire que nous avons moins de performance si nous appliquons le modèle sur un roman complet qui comprend beaucoup plus de phrases non géographiques que de phrases géographiques. La solution de cette limitation peut être une des perspectives possibles de notre méthode, c'est-à-dire en entraînant notre modèle sur des corpus de différentes structures. Une autre perspective possible repose sur le fait de tester des modèles pré-entraînés spécifiquement pour le français (ex: FlauBERT).

Ce stage constitue une première expérience en tant que stagiaire dans un laboratoire de recherche français, ce qui me permet d'observer des expériences nouvelles pour mon parcours professionnel et personnel. Durant ce stage en tant que stagiaire, mon expérience a connu de nombreux points positifs. Tout d'abord, le suivi de la progression dans mon travail a été permis grâce à une très bonne communication avec mon encadrant, Monsieur Ludovic Moncla. Nous effectuions un point d'avancement du travail une fois par semaine, ce qui m'a permis d'améliorer la compréhension de chaque petite

[17] https://gitlab.liris.cnrs.fr/HExtGEO/sentence-classification

partie dans le projet. Ensuite, ce stage m'a donné l'opportunité d'améliorer mes compétences en communication, en présentation orale du travail effectué et en esprit de synthèse. En effet, de nombreuses réunions ont eu lieu au cours de ces quelques mois, notamment deux réunions avec la direction du laboratoire, et des rendez-vous hebdomadaires avec l'équipe et les stagiaires du laboratoire. Ces rendez-vous m'ont permis de prendre connaissance du travail de mes collègues, et de m'entrainer à présenter mes recherches, à structurer ma pensée et à organiser mes idées. Ce travail d'expression orale a été l'occasion de m'exercer pour la présentation lors de la soutenance. Les réunions à distance et le regroupement des stagiaires dans le même bureau au début du stage m'ont également permis de m'intégrer facilement dans l'équipe des stagiaires du laboratoire de recherche. De plus, la communication régulière avec monsieur Moncla, sa présence et la rapidité de ses retours m'ont permis d'avancer à un rythme soutenu dans la progression de mon travail.

J'ai également eu l'opportunité d'améliorer mes compétences scientifiques, grâce à la recherche et la rédaction d'un état de l'art, mais aussi mes compétences théoriques comme celles sur l'apprentissage automatique, l'apprentissage profond et le domaine de TALN (Traitement Automatique du Langage Naturel). J'ai également eu l'occasion de perfectionner mes compétences techniques grâce à l'utilisation d'outils et d'environnements divers pour réaliser les tâches qui m'ont été confiées. Enfin, malgré la situation sanitaire que nous avons traversé lors de ces quelques mois de stage, j'ai eu l'opportunité de développer mes compétences organisationnelles en travaillant en autonomie et en organisant mon propre planning. Le travail en autonomie m'a permis d'être responsable de l'avancée de mon travail. Je suis désormais capable de hiérarchiser et de visualiser plus facilement et les tâches qu'il me reste à accomplir.

J'ai cependant rencontré quelques difficultés au cours de mon stage, notamment dues à la situation sanitaire. En effet, l'une des difficultés auxquelles j'ai dû faire face est la réorganisation du travail notamment à cause du contexte imposé par le confinement. J'ai ainsi dû apprendre à travailler davantage en autonomie, seul et à distance, ce qui m'a amené à revoir mes horaires de travail et à arranger mon planning. Enfin, le fait de travailler en télétravail m'a privé du contact direct avec mon maître de stage et m'a demandé d'adapter nos rencontres aux outils de communication numérique.

# Bibliographie


Alemi, A. A., & Ginsparg, P. (2015). Text Segmentation based on Semantic Word Embeddings. *ArXiv:1503.05543 [Cs]*. http://arxiv.org/abs/1503.05543

Baranes, M. (2012). *Vers la correction automatique de textes bruités : Architecture générale et détermination de la langue d'un mot inconnu*. 15.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised Sequence Learning. *arXiv:1511.01432*

*[cs]*. http://arxiv.org/abs/1511.01432

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv:1810.04805 [Cs]*. http://arxiv.org/abs/1810.04805

Hochreiter, S., & Schmidhuber, J. (1997, décembre). *(16) (PDF) Long Short-term Memory*.ResearchGate.https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

Jelodar, H., Wang, Y., Yuan, C., & Feng, X. (2017, novembre). *(16) (PDF) Latent Dirichlet Allocation (LDA) and Topic modeling : Models, applications, a survey*. ResearchGate. https://www.researchgate.net/publication/321069759_Latent_Dirichlet_Allocation_LDA_and_Topic_modeling_models_applications_a_survey

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746-1751. https://doi.org/10.3115/v1/D14-1181

Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv:1609.02907 [Cs, Stat]*. http://arxiv.org/abs/1609.02907

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., & Fidler, S. (2015). Skip-Thought Vectors. *ArXiv:1506.06726 [Cs]*. http://arxiv.org/abs/1506.06726

Le, Q., & Mikolov, T. (2013). *Distributed Representations of Sentences and Documents*. 9.

Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec : Learning Generic Context Embedding with Bidirectional LSTM. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 51-61. https://doi.org/10.18653/v1/K16-1006

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. 9.

Pennington, J., Socher, R., & Manning, C. (2014). Glove : Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532-1543. https://doi.org/10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *ArXiv:1802.05365 [Cs]*. http://arxiv.org/abs/1802.05365

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained Models for Natural Language Processing : A Survey. *ArXiv:2003.08271 [Cs]*. http://arxiv.org/abs/2003.08271

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language*

*Understanding by Generative Pre-Training*. 12.

Si, Y., Wang, J., Xu, H., & Roberts, K. (2019). Enhancing Clinical Concept Extraction with Contextual Embeddings. *Journal of the American Medical Informatics Association*, *26*(11), 1297-1304. https://doi.org/10.1093/jamia/ocz096

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. 12.

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2020). How to Fine-Tune BERT for Text Classification? *ArXiv:1905.05583 [Cs]*. http://arxiv.org/abs/1905.05583

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *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)*, 1556-1566. https://doi.org/10.3115/v1/P15-1150

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs]*. http://arxiv.org/abs/1706.03762

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation. *ArXiv:1609.08144 [Cs]*. http://arxiv.org/abs/1609.08144

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G. H., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2019). Multilingual Universal Sentence Encoder for Semantic Retrieval. *ArXiv:1907.04307 [Cs]*. http://arxiv.org/abs/1907.04307