UNIVERSITÉ DU QUÉBEC À MONTRÉAL

SUR LE PROBLÈME DU DICTIONNAIRE INVERSÉ

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INTELLIGENCE ARTIFICIELLE

PAR

JULIEN GUITÉ-VINET

AOÛT 2023

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

ALBERT A Lite BERT.

API Application Programming Interface.

BERT Bidirectional Encoder Representations from Transformers.

BiLSTM Bidirectional Long Short Term Memory.

BoW Bag of Word.

BPE Byte Pair Encoding.

CFNN Cascade Feed-Forward Neural Network.

CNN Convolution Neural Network.

DeBERTa Decoding-enhanced BERT with Distangled Attention.

FNN Feed-Forward Neural Network.

GPT Generative Pretrained Transformer.

GPU Graphic Processor Unit.

IDF Inverse Document Frequency.

JJ Adjectif.

KB Knowledge Base.

LSTM Long Short Term Memory.

LCA Least Common Ancestor.

MNIST Modified National Institute of Standards and Technology database.

MSE Mean square error.

NN Nom commun singulier.

NP Phrase nominale.

POS Part-of-speech.

PSC Perceptron simple couche.

RoBERTa Robustly Optimized BERT Pretraining Approach.

RB Adverbe.

RNN Recurrent Neural Network.

TAL Traitement Automatique du Language.

TALN Traitement Automatique du Language Naturel.

TF Term Frequency.

UQAM Université du Québec à Montréal.

VP Phrase verbale.

VB Verbe à sa forme de base.

VBG Verbe au participle présent.

VBN Verbe au participle passé.

NOTATION

Variables

w tout symbole répertorié dans un dictionnaire donné.

b jeton provenant de la segmentation d'un symbole w.

h état caché d'un réseau de neurones.

ℓ la profondeur d'une couche d'un réseau de neurones.

Vecteurs

v_w représentation vectorielle d'un symbole w.

e_w représentation vectorielle du plongement de jeton d'un symbole w.

Matrices

W matrice de poids.

E matrice de plongements de jetons.

Ensembles

P suite ordonnée d'un ou plusieurs symboles, notée $\langle w_1, w_2, \ldots, w_i, \ldots w_n \rangle$.

D un dictionnaire défini comme un ensemble de paires (w, P) où w est un symbole et P est une définition.

V ensemble de symboles uniques dans un dictionnaire.

Θ paramètres d'un modèle.

Fonctions

$m(w)$ fonction retournant l'ensemble de toutes les définitions de w se retrouvant dans les paires (w, P) contenu dans D pour un mot w.

$m^{-1}(w)$ fonction d'association inverse de m retournant l'ensemble des définitions contenant w dans leurs définitions P, c'est-à-dire $m^{-1}(w) = \{P \mid \text{il existe } (w', P) \in D \text{ avec } w \in P\}$.

$p(w \mid x)$ fonction retournant la probabilité conditionnelle qu'un symbole w soit choisit quand l'événement x survient.

$\mathrm{T}(P)$ fonction retournant l'arbre d'analyse syntaxique pour une phrase P.

$M(P)$ fonction d'évaluation d'un modèle appliquée à la phrase P.

J fonction de coût quantifiant la précision des prévisions d'un modèle.

ϕ fonction d'activation, par exemple $\phi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ pour tout $x \in \mathbb{R}$.

σ fonction sigmoïde $= \sigma(x) = \frac{1}{1+e^{-x}}$ pour tout $x \in \mathbb{R}$.

RÉSUMÉ

Un dictionnaire est un système de symboles qui associe des mots à d'autres mots de telle sorte que la signification des mots définis est établie à travers les mots des définitions. Le problème du dictionnaire inversé consiste à deviner un mot à partir d'une définition. Résoudre ce problème s'avère utile dans plusieurs contextes, puisque cela permet de représenter la connaissance lexicale, d'attribuer des mots-clés à un ensemble d'idées ou de dissiper l'ambiguïté sémantique autour de concepts.

Des travaux récents en traitement du langage naturel proposent des modèles de type transformeur (BERT, GPT-4) qui sont entraînés sur un grand corpus de texte et qui sont ensuite réglés spécifiquement afin d'accomplir une tâche propre au domaine. Leur entraînement pose un défi parce qu'elle peut, entre autres, être coûteuse en calcul, difficile à paramétrer, et dépend de la taille et de la qualité des ensembles de données utilisés durant leur entraînement.

Ce mémoire aborde les tâches et problématiques associées au traitement automatique du langage et les différents modèles et stratégies permettant de résoudre le problème du dictionnaire inversé. Nos travaux explorent de possibles usages que l'on peut faire des transformeurs dans le cadre de cette problématique. Les modèles développés sont ensuite utilisés dans le cadre d'un jeu sérieux appelé *Jeu du dictionnaire*. Plus globalement, les résultats obtenus pointent vers une exploration plus approfondie des modèles de langue profonds ainsi que le transfert d'apprentissage afin de résoudre d'autres types de tâches.

INTRODUCTION

La linguistique informatique est un domaine interdisciplinaire concerné par la modélisation informatique du langage, ainsi que par l'étude d'approches informatiques appropriées aux questions linguistiques. Le traitement automatique du langage (TAL) est un sous-domaine de la linguistique, de l'informatique et de l'intelligence artificielle qui s'intéresse aux interactions entre les ordinateurs et le langage humain, en particulier comment programmer des ordinateurs pour traiter et analyser des données plus ou moins structurées et principalement exprimées en langage naturel. L'objectif consiste donc à faire « comprendre » à un ordinateur le contenu des documents, y compris les nuances contextuelles et sémantiques. La technologie peut alors extraire, organiser ou catégoriser les informations et les idées contenues dans les documents.

On atteint cet objectif en accomplissant des tâches dans un spectre de difficultés variables qui comprend le traitement de texte, l'analyse morphologique, l'analyse syntaxique, l'extraction de la sémantique lexicale, de la sémantique relationnelle et de l'analyse du discours. Certaines de ces tâches ont des applications directes dans le monde réel, tandis que d'autres sont utilisées dans le but de résoudre des tâches de niveau supérieur plus complexes comme la traduction automatique et la génération de texte.

Les premiers systèmes de traitement du langage étaient principalement basés sur des systèmes experts, implémentés à l'aide d'un codage manuel d'un ensemble de règles et d'heuristiques, couplé à une recherche dans un dictionnaire (Weizenbaum, 1966; Styne, 1986; Lesk, 1986; Pollard et Sag, 1988). Généralement, les systèmes basés sur des règles écrites par un expert du domaine gagnent en précision en augmentant leur complexité. Cependant, il existe une limite à les développer, au-delà de laquelle les systèmes deviennent difficiles à maintenir, car les règles sont complexes informatiquement et elles demandent l'intervention d'experts (Partridge, 1987).

Pour contourner les inconvénients des systèmes experts, certains travaux (voir (Rumshisky et Stubbs, 2017) pour une revue détaillée récente) se sont concentrés sur l'apprentissage automatique. Le paradigme de l'apprentissage automatique appelle à utiliser l'inférence statistique pour apprendre automatiquement de telles règles grâce à l'analyse de grands corpus. Certains outils statistiques, tels que les arbres de décisions et les modèles de Markov, basés sur la théorie de l'information, ont été

proposés (Shannon, 1951; Abney et Light, 1999; Phu *et al.*, 2017). Ceux-ci nécessitent généralement une ingénierie élaborée de la linguistique computationnelle, qui repose sur des sous-tâches, telles que la segmentation des mots, l'analyse syntaxique, la lemmatisation, la *stemmatisation* et l'étiquetage de la partie du discours (Gage, 1994; Shaw *et al.*, 2013; Kandola *et al.*, 2002).

Depuis quelques années, on remarque un engouement pour l'utilisation des réseaux de neurones profonds, qui se reflète dans la littérature scientifique du domaine (Huang *et al.*, 2017; Radford *et al.*, 2018; Devlin *et al.*, 2019; Lan *et al.*, 2019). De tels modèles se révèlent plus robustes lorsqu'ils reçoivent des entrées inconnues, en particulier des entrées contenant des erreurs, comme c'est courant pour les données du monde réel (Yang *et al.*, 2019; Xue *et al.*, 2021). Ils présentent des résultats plus fiables lorsqu'ils sont intégrés dans un système plus vaste comprenant plusieurs sous-tâches (Devlin *et al.*, 2019; Zhang *et al.*, 2020). Une technique populaire afin de résoudre ces tâches repose sur le plongement des mots, c'est-à-dire une représentation vectorielle qui capture en grande partie leurs propriétés sémantiques. Toutefois, les réseaux de neurones nécessitent dans la plupart des cas une quantité importante de mémoire et de données étiquetées (Huang *et al.*, 2017). Ils impliquent aussi une phase d'apprentissage de longue durée et sont difficiles à paramétrer (Radford *et al.*, 2018; Liu *et al.*, 2019).

Plusieurs tâches reposent sur la notion de similarité sémantique des concepts, qui est bien étudiée dans la littérature (Chen et Goodman, 1999; Achananuparp *et al.*, 2008; Niwattanakul *et al.*, 2013). Une tâche connexe à la désambiguïsation du sens des mots concerne le problème du dictionnaire inversé. Dans sa forme la plus simple, un dictionnaire de langue usuelle, comme le Petit Larousse ou le Merriam Webster's, peut être vu comme une association entre le sens d'un mot et sa définition. Le problème du dictionnaire inversé s'intéresse plus spécifiquement à l'association inverse, c'est-à-dire qu'on doit deviner un mot à partir de sa définition (Shaw *et al.*, 2013; Kartsaklis *et al.*, 2020).

Un dictionnaire attribue généralement plusieurs sens à un mot, chacun d'entre eux ayant une signification différente en fonction du contexte dans lequel le mot est employé dans une phrase. Parallèlement, on appelle homographes les mots qui s'écrivent de la même manière, peu importe leur prononciation, par exemple :

— Je *vis* ces *vis*

— Tu *as* un *as* dans ton jeu

— Il *est* né à l'*est*

— *Rose* arrose la *rose rose*

Un modèle linguistique pouvant résoudre efficacement le problème du dictionnaire inversé doit donc être en mesure de capturer les concepts de synonymie, d'antonymie, de polysémie, d'homographie, ainsi que les multiples relations sémantiques contextuelles. Cela est d'autant plus difficile qu'il s'agit d'une tâche de classification sur la taille du vocabulaire. Or, dans la langue française, par exemple, on dénombre plus de 100 000 mots dont plusieurs sont polysémiques, c'est-à-dire qu'ils possèdent de multiples définitions.

Trouver une solution à ce problème s'applique à plusieurs contextes : la représentation de la connaissance, l'attribution de mot-clé et la désambiguïsation sémantique, particulièrement pour le traitement du langage naturel, la recherche d'informations, la modélisation du langage et les systèmes de bases de données (Shaw *et al.*, 2013). Cela peut aussi s'avérer utile en support à d'autres tâches plus complexes comme la rédaction et la traduction (Miller *et al.*, 1993; Hill *et al.*, 2016). D'autre part, cela s'inscrit dans un contexte plus large qui englobe le concept de lexique mental (Croft *et al.*, 2004). Le lexique mental diffère du simple lexique puisqu'« il n'est pas simplement un ensemble général de mots, au lieu de cela, il traite de la façon dont ces mots sont activés, traités et récupérés par chaque locuteur » (Elman, 2012).

Pour résoudre la tâche du dictionnaire inversé, une stratégie relativement simple consiste à prendre une définition comme entrée, y inclure les synonymes des mots de l'entrée et ensuite dénombrer l'occurrence des mots de l'entrée pour chacune des définitions d'un dictionnaire. On choisit le mot associé à la définition ayant dénombré le plus de correspondances (Crawford et Crawford, 1997; El-Kahlout et Oflazer, 2004). L'originalité de ce type de système repose sur le traitement des définitions et sur l'inclusion des synonymes parmi des ressources variées.

Une autre stratégie semblable consiste à utiliser un sac de mots (BoW pour *bag-of-words*), c'est-à-dire une modélisation vectorielle basée sur le nombre d'occurrences des mots dans un document (Kandola *et al.*, 2002; Bilac *et al.*, 2004). L'axe de recherche pour ce genre de méthode est basé sur l'expérimentation des mesures de quantification des mots ainsi que sur des mesures de similarité des

représentations vectorielles.

Dans un dictionnaire contenant plus d'une centaine de milliers mots, il faut au minimum autant de comparaison pour retourner le mot associé à la définition ayant la similarité la plus élevée pour une phrase donnée. Afin de réduire l'espace de recherche, on utilise des méthodes statistiques et des arbres d'analyse pour emmagasiner dans des bases de données ou des graphes sémantiques des informations utiles afin de produire des requêtes qui conduiront au mot cible (Dutoit et Nugues, 2002; Zock et Bilac, 2004; Shaw *et al.*, 2013; Thorat et Choudhari, 2016; Siddique et Beg, 2022). La contribution de ces études repose sur la complexité des arbres d'analyse et de la recherche d'une mesure de qualification des liens sémantiques entre les mots.

Pour mesurer la précision des prévisions d'un modèle monolingue anglophone, Hill et al. ont construit un ensemble de données qui sont tirées principalement de dictionnaires anglophones et qui sert de référence d'évaluation (en anglais, *benchmark*) (Hill *et al.*, 2016). Des stratégies basées sur des réseaux de neurones artificielles se sont avérées donner des résultats davantage précis lorsqu'elles sont testées sur ces données. Par exemple, étant donné un corpus de texte, l'algorithme Word2vec fixe les paramètres d'un modèle neuronal de manière à maximiser la probabilité qu'un mot soit associé à son contexte (Mikolov *et al.*, 2013a). De cette manière, on calcule un plongement pour chacun des mots du vocabulaire du corpus. Une fois les paramètres fixés, on peut combiner les plongements de chaque mot d'une définition dans le but d'obtenir une estimation du vecteur cible (Hill *et al.*, 2016).

Comme mentionné auparavant, le modèle résolvant le problème du dictionnaire inversé doit idéalement assimiler les concepts sémantiques reliant les mots et les phrases entre eux. Les réseaux neuronaux récurrents tels que les RNN (Rumelhart *et al.*, 1985), les réseaux LSTM (Hochreiter et Schmidhuber, 1997) et les transformeurs (Vaswani *et al.*, 2017; Radford *et al.*, 2018; Devlin *et al.*, 2019; Lan *et al.*, 2019) ont démontré être suffisamment expressifs pour les prendre en compte.

Dans le contexte du problème du dictionnaire inversé, un réseau de neurones prend en entrée une représentation vectorielle des mots d'une définition de dictionnaire. Les réseaux récurrents traitent ces représentations l'une à la suite de l'autre en prenant en compte les représentations auparavant traitées. Récursivement, le réseau met à jour des états internes. L'état final devient un résumé des

informations contenues dans la phrase d'entrée.

Tout comme les réseaux récurrents, les transformeurs semblent adaptés afin de capturer les relations sémantiques reliant les mots d'une phrase. L'architecture originale d'un transformeur comprend une pile d'encodeurs et une pile de décodeurs de même taille (Vaswani *et al.*, 2017). Les entrées passent par une couche d'auto-attention, c'est-à-dire une couche qui aide l'encodeur à se concentrer sur d'autres mots dans la phrase d'entrée lorsqu'il encode un mot spécifique. La sortie de la pile d'encodeurs est ensuite transmise aux décodeurs et ceux-ci propagent leurs résultats de décodage jusqu'à ce qu'un symbole spécial soit atteint par la dernière couche indiquant que le transformeur a terminé le traitement.

En se comparant à d'autres réseaux de neurones, pour plusieurs tâches du TAL, les transformeurs se sont avérés produire de meilleurs résultats avec des marges significatives (Radford *et al.*, 2018; Yang *et al.*, 2019; Devlin *et al.*, 2019). À la lumière de ces résultats, ceux-ci semblent tout indiqués pour résoudre le problème du dictionnaire inversé. Plusieurs variantes de transformeurs ont vu le jour et, dans ce mémoire, nous explorons différents traitements du problème par le biais des transformeurs.

Le premier chapitre plonge dans la problématique et les diverses tâches associées au TAL, plus précisément les tâches associées au traitement du texte, à l'analyse morphologique, syntaxique et lexicale. On y décrit le problème du dictionnaire inversé ainsi que les notations mathématiques qui sont utilisées tout au long du mémoire. On y présente aussi un jeu sérieux appelé le *Jeu du dictionnaire* (Blondin Massé *et al.*, 2008; Guite-Vinet *et al.*, 2023).

Le deuxième chapitre traite d'approches classiques permettant de résoudre le problème du dictionnaire inversé. On y détaille l'approche basée sur les sacs de mots et la distance cosinus (Rumshisky et Stubbs, 2017), ainsi qu'une approche statistique, basée sur des arbres d'analyse de phrase et la méthode hongroise, permettant de résoudre le problème en question (Kuhn, 1955; Shaw *et al.*, 2013). On développe aussi différentes approches afin de représenter de vectoriellement les mots, et plus précisément l'encodage *one-hot*, la méthode statistique TF-IDF, les matrices de cooccurrence et les plongements de mots, dont l'algorithme Word2vec.

Le troisième chapitre s'intéresse aux différentes approches neuronales pouvant mener à des modèles résolvant le problème du dictionnaire inversé. Dans cette optique, on précise comment les projections

linéaires des plongements de mots peuvent être utilisées dans des réseaux récurrents RNN et LSTM combinés à des tâches auxiliaires.

Le quatrième chapitre détaille le fonctionnement des transformeurs en général et plus en détail les modèles BERT et GPT. BERT (de l'anglais, *Bidirectional Encoder Representations from Transformers*) est un modèle de langue profond basé sur les transformeurs et inventé par Google (Devlin *et al.*, 2019). GPT (de l'anglais, *Generative Pre-trained Transformer*) est un modèle de langue profond autorégressif, aussi basé sur les transformeurs et qui a été inventé par Open-Ai (Radford *et al.*, 2018). D'abord préentraînés, ces modèles peuvent être affinés afin de résoudre une variété de tâches qui pourraient ne pas être associées aux tâches sur lesquels ils ont été formés.

Le cinquième chapitre présente une expérience qui a été menée portant sur l'utilisation de transformeurs afin de résoudre le problème du dictionnaire inversé et discute des résultats obtenus.

CHAPITRE 1
NOTIONS PRÉLIMINAIRE

L'objectif de ce chapitre est de décrire diverses tâches associées au TAL, plus précisément les tâches associées au traitement du texte, à l'analyse morphologique, syntaxique et lexicale. Il a pour but d'établir le cadre théorique dans lequel s'inscrit le mémoire.

1.1 Traitement automatique du langage naturel

Les défis dans le domaine du traitement du langage naturel impliquent souvent la reconnaissance, la compréhension ou la génération du langage naturel. Bien que certaines tâches se concentrent sur l'analyse de la parole ou du son, par exemple la reconnaissance vocale (Morise *et al.*, 2016), nous limitons la portée du mémoire aux tâches concernant exclusivement le texte. Afin de bien cerner la complexité de la linguistique computationnelle, nous divisons en six catégories certaines des tâches du traitement automatique du langage naturel soit : (**i**) le traitement du texte, (**ii**) l'analyse morphologique, (**iii**) l'analyse syntaxique, (**iv**) la sémantique lexicale, (**v**) la sémantique relationnelle et (**vi**) l'analyse discours.

1.1.1 Traitement du texte

Les tâches de traitement du texte transforment ou exploitent un texte dans un but précis.

Reconnaissance de caractères. Étant donné une image représentant un texte imprimé, on cherche à déterminer le texte correspondant (Kahan *et al.*, 1987). Par exemple, la base de données MNIST (en anglais, *Modified National Institute of Standards and Technology database*) est une base de données de chiffres manuscrits couramment utilisée pour entraîner des modèles de traitement d'images (LeCun, 1998).

Segmentation de phrases. La segmentation des phrases est un problème de division d'une chaîne de caractères en ses phrases composantes. En français, cela peut sembler assez simple puisqu'il est possible de démarquer les phrases à partir de la ponctuation. Cependant, ce problème n'est pas

si anodin en raison des exceptions, par exemple l'utilisation des caractères de point d'abréviations (Reynar et Ratnaparkhi, 1997).

Segmentation de mots. La segmentation de mots (en anglais, *tokenization*) consiste à séparer un morceau de texte en mots séparés. Pour une langue comme l'anglais, c'est relativement simple, puisque les mots sont généralement séparés par des espaces. Cependant, certaines langues écrites, comme le chinois ou le japonais, ne marquent pas les frontières des mots de cette manière. Dans ces langues, la segmentation du texte est une tâche importante nécessitant une connaissance du vocabulaire et de la morphologie des mots dans la langue (Palmer, 2000).

Une solution plus générale à cette situation repose sur l'encodage BPE (*Byte Pair Encoding*), qui consiste à compresser les données en subdivisant les mots en jetons afin de créer un vocabulaire (Gage, 1994). L'intuition derrière cette idée est que les paires de symboles consécutifs les plus courantes sont remplacées (ou fusionnées) par un symbole qui n'apparaît pas dans ces données. Par exemple, considérons des données devant être encodées sous la forme du mot `aabdaaabac`. Puisque la paire de symboles `aa` apparaît le plus souvent, on la remplace par `Z`, car `Z` n'apparaît pas dans ces données. On obtient `ZabdZabac` avec `Z=aa`. La prochaine paire de symboles communs est `ab` que l'on remplace par `Y`. On a maintenant `ZYdZYac` où `Z=aa` et `Y=ab`. La seule paire de symboles restante est `ac`. Celle-ci n'apparaît qu'une seule fois et n'est donc pas encodée. On encode ensuite récursivement `ZY` en tant que `X`. Les données sont maintenant transformées en `XdXac` où `X=ZY`, `Y=ab` et `Z=aa`. Les données ne peuvent pas être compressées davantage, car il n'y a pas de paires de symboles apparaissant plus d'une fois. Les données sont ensuite décodées en effectuant des remplacements dans l'ordre inverse. L'encodage BPE garantit que les mots les plus courants sont représentés dans le vocabulaire sous la forme d'un seul jeton tandis que les mots rares sont décomposés en deux ou plusieurs jetons de sous-mots (Khanna, 2021).

WordPiece (Wu *et al.*, 2016) est un algorithme similaire à BPE. Mais contrairement à BPE, WordPiece ne fusionne pas la paire de symboles la plus fréquente, mais plutôt celle qui maximise la vraisemblance (en anglais, *likelihood*) du corpus. Par exemple, comment choisir si le mot `bleu` doit être encodé avec `(bl,e,u)` ou `(b,le,u)` ou même `(b,l,eu)` ? Intuitivement, WordPiece évalue ce qu'il perd en fusionnant deux symboles pour s'assurer que cela en vaut la peine.

D'ailleurs, la taille du vocabulaire de base et le nombre de fusions possibles sont des paramètres à définir. Par exemple, le transformeur GPT (voir section 4.3) a une taille de vocabulaire égale à 40 478, car on introduit 478 caractères de base et 40 000 fusions possibles. De plus, BPE place le caractère `@` à la fin des jetons tandis que WordPiece place les caractères `##` au début. Par exemple, le mot *marcher* peut être segmenté par BPE avec (`march@`, `er`), tandis que WordPiece le ferait avec (`march`,`##er`).

Mots fonctionnels. Les mots fonctionnels (en anglais, *stop words*) sont des mots qui sont filtrés avant ou après le traitement du texte. Les mots fonctionnels font référence à des mots qui n'ajoutent pas beaucoup de valeur sémantique à des phrases telles que *et*, *le* et *un*. Il n'existe pas de liste universelle unique de mots fonctionnels. Étant donné que ces mots se retrouvent en grande quantité, un peu partout dans les documents, et qu'ils n'ont pas beaucoup de valeurs sémantiques, on les supprime souvent puisqu'ils peuvent ajouter du bruit au traitement et prendre de l'espace mémoire. Notons que la liste des mots considérés comme fonctionnels peut changer en fonction de son application (Wilbur et Sirotkin, 1992).

1.1.2 Analyse morphologique

La morphologie découpe les mots en unités significatives, en se basant non seulement sur leur rôle grammatical, mais aussi sur leurs possibilités de combinaison avec d'autres unités linguistiques (Léon et Bhatt, 2017). Les monèmes [1] résultent d'un découpage particulier des informations grammaticales opérées par le système linguistique. Les monèmes se divisent en deux catégories :

1. les *lexèmes*, qui servent à exprimer des informations de type conceptuel, c'est-à-dire l'action, l'idée et l'objet en cause
2. les *morphèmes*, qui servent à exprimer des distinctions grammaticales, par exemple, le genre, le nombre, le temps, et le mode

Les morphèmes, comme les lexèmes, se composent d'éléments de sens plus petits encore que l'on appelle les sèmes grammaticaux. Les sèmes grammaticaux sont les distinctions grammaticales établies par la langue.

Prenons, par exemple, la phrase « Pauline et François travaillaient ». Le morphème *aient* exprime

1. En linguistique, unité significative minimale (Robert, 2022).

plusieurs distinctions morphologiques par les sèmes grammaticaux suivants : la personne (première/deuxième/troisième), le nombre (singulier/pluriel), le temps (passé/présent/futur), le mode (indicatif/conditionnel/subjonctif), l'aspect (achevé/inachevé) et la voix (active/passive). On appelle ce type de morphème une flexion verbale (Léon et Bhatt, 2017).

Certains morphèmes, par exemple *es* et *ées*, sont relatifs aux catégories du nom et de l'adjectif. Ils sont appelés des flexions nominales et adjectivales. Ils expriment deux types de distinction morphologique, soit le genre et le nombre (Léon et Bhatt, 2017).

Lemmatisation. Dans une langue comme le français, un lemme peut être :

- simple : un seul mot comme *amour*
- composé : un mot composé comme *rouge-gorge*
- complexe :
  - un syntagme [2] comme *grand-chose*
  - une expression complète comme *je-ne-sais-quoi* (Said El Hassani, 2011)

La lemmatisation est le processus qui consiste à réduire les mots à une forme normalisée. En français, la tâche consiste essentiellement à tronquer un mot de façon à retourner son lemme, c'est-à-dire une forme de base du mot tirée du dictionnaire (Balakrishnan et Ethel, 2014).

Segmentation morphologique. Le but est de séparer les mots en morphèmes [3] individuels et identifier la classe des morphèmes (Dasgupta et Ng, 2007). La difficulté de cette tâche dépend en grande partie de la complexité de la morphologie de la langue à l'étude.

En français, étant donné sa morphologie flexionnelle, il est souvent possible de faire la segmentation morphologique en modélisant toutes les formes possibles d'un mot. Dans des langues telles que le turc, une telle approche n'est pas possible, car chaque entrée du dictionnaire a des milliers de formes possibles (Oflazer *et al.*, 1994).

Étiquetage de la partie du discours. Dans le cas de la langue française, la grammaire traditionnelle classe les mots en neuf catégories, appelées également *parties du discours* (en anglais, POS

2. [...]groupe d'éléments formant une unité dans une organisation hiérarchisée (Larousse, 2022).

3. Forme minimum douée de sens (mot simple ou élément de mot) (Robert, 2022).

pour *part-of-speech*) : adjectif, adverbe, conjonction, déterminant, interjection, nom, préposition, pronom et verbe (Léon et Bhatt, 2017). Étant donné une phrase, on doit déterminer le POS de chaque mot. La difficulté réside dans le fait que de nombreux mots peuvent avoir différents POS attribués. Par exemple, généralement le mot *dîner* est un verbe, mais dans la phrase « Je prépare le dîner », il s'agit d'un nom (Tseng *et al.*, 2005).

Extraction de la racine. L'extraction de la racine (en anglais, *stemming*) est le processus de troncature des mots à une forme de base. Par exemple, *ferm* est la racine de *fermé, fermeture, fermant*. L'extraction de la racine donne des résultats similaires à la lemmatisation, mais on utilise une base de règles au lieu d'un dictionnaire (Balakrishnan et Ethel, 2014).

En anglais, la variation morphologique se produit à l'extrémité droite d'un mot, ce qui a motivé l'extraction de la racine basée sur un dictionnaire de suffixes par troncature à droite (Lovins, 1968; Sproat et Wilkes, 1992). Il s'agit d'une approche simple pouvant introduire des troncatures excessives ou des sous-troncatures. Une troncature excessive se produit lorsque l'opération de troncature retourne une racine trop courte. Cela peut mener à des résultats indésirables, comme pour les mots *medical* et *media* qui pourraient partager la même racine *med*. À l'inverse, une sous-troncature se produit si l'opération de troncature extrait une racine trop longue. Cela peut occasionner des mots sémantiquement connexes à être définies par différentes racines, comme en tronquant *bibliographiquement* avec *bibliographique*, plutôt qu'avec *bibliograph*, c'est-à-dire une racine qui engloberait également *bibliographie*. Afin d'éviter ces erreurs, une série de règles contextuelles peut être fournie dans le but d'ajuster les troncatures de suffixes. Par exemple, en contraignant une racine à contenir une longueur minimale de deux caractères, cela empêche le suffixe *ette* d'être supprimé au mot *dette*.

Porter utilise plutôt une mesure basée sur le nombre de chaînes consonne-voyelle-consonne restantes après la suppression d'un suffixe (Porter, 1980). L'utilisation d'environ un cinquième des suffixes répertoriés dans le dictionnaire de Lovins est suffisante pour une extraction de la racine efficace, puisque l'algorithme de Porter est de nature itérative, c'est-à-dire qu'on tronque les suffixes par étapes. L'algorithme comporte cinq étapes : **(i)** gérer les suffixes flexionnels, **(ii,iii,iv)** traiter les suffixes dérivationnels, **(v)** recoder.

1.1.3 Analyse syntaxique

En linguistique, la syntaxe s'occupe des règles par lesquelles les unités linguistiques se combinent en phrases. L'analyse syntaxique a trois buts principaux :

— définir la structure des groupes de catégories de mots

— établir la structure des phrases

— décrire les interprétations possibles des différentes structures syntaxiques (Léon et Bhatt, 2017).

Désambiguïsation des limites de la phrase. Étant donné un texte, on veut trouver les limites de la phrase. Les limites des phrases sont souvent marquées par des points ou d'autres signes de ponctuation, mais ces mêmes caractères peuvent servir à d'autres fins, comme pour marquer des abréviations (Reynar et Ratnaparkhi, 1997).

Arbre d'analyse. On cherche à représenter sous forme arborescente la structure grammaticale d'une phrase donnée (Vauquois, 2003; Bengtson et Roth, 2008). La grammaire des langues naturelles est ambigüe et les phrases peuvent avoir plusieurs analyses possibles. Il existe deux principaux types d'analyse : l'analyse de dépendance et l'analyse de circonscription. L'analyse de dépendance se concentre sur les relations entre les mots dans une phrase, marquant des choses comme les objets primaires et les prédicats. L'analyse de circonscription se concentre sur la construction de l'arbre d'analyse à l'aide d'une grammaire probabiliste sans contexte (Zhang, 2020).

Par exemple, étant donné une phrase d'entrée, OpenNLP produit un arbre d'analyse de dépendance, où les mots les plus significatifs quant au sens de la phrase apparaissent plus haut que les mots qui le sont moins (Foundation, 2009). La figure 1.1 illustre un tel arbre pour la phrase « Le ciel est bleu ».

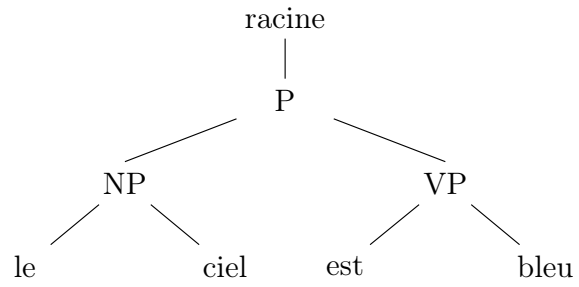

FIGURE 1.1 – Pour la phrase « le ciel est bleu », OpenNLP produirait un arbre d'analyse où P, VP et NP désigneraient respectivement Phrase, Phrase verbale et Phrase nominale.

1.1.4 Sémantique lexicale

La sémantique lexicale s'intéresse au sens des mots individuels dans leur contexte. On cherche à voir quelle est la signification des mots individuels dans leur contexte (Léon et Bhatt, 2017).

Désambiguïsation du sens des mots. La désambiguïsation du sens des mots est le processus d'identification du sens d'un mot dans une phrase ou un autre segment de contexte (Stevenson et Wilks, 2003). En linguistique informatique, il s'agit d'un problème difficile qui affecte d'autres domaines liés à l'informatique, tels que l'amélioration de la pertinence des moteurs de recherche et la résolution des anaphores [4](Uryupina *et al.*, 2011). Identifier un sens particulier d'un mot donné dans son contexte est difficile, car le modèle linguistique doit être en mesure de capturer les concepts de synonymie [5], de polysémie [6], d'homographie [7], ainsi que les multiples relations sémantiques, notamment l'hyponymie [8], l'hyperonymie [9], l'antonymie et la métonymie [10].

4. Processus par lequel un segment du discours (dit anaphorique) renvoie à un autre segment (dit antécédent) apparu dans le même contexte.[...] (Larousse, 2022)

5. Relation qu'entretiennent entre eux divers termes ou expressions ayant le même sens ou un sens voisin (Larousse, 2022).

6. Propriété d'un terme qui présente plusieurs sens (Larousse, 2022).

7. Se dit d'homonymes ayant la même forme graphique (Larousse, 2022).

8. Rapport d'inclusion entre des unités lexicales, considéré comme orienté du plus spécifique au plus général (Larousse, 2022).

9. Terme dont le sens inclut le sens d'un ou plusieurs autres termes, qui sont ses hyponymes (Larousse, 2022).

10. Phénomène par lequel un concept est désigné par un terme désignant un autre concept qui lui est relié par une relation nécessaire (Larousse, 2022).

Plusieurs travaux ont été effectués en utilisant WordNet comme référence lexicale à des fins de désambiguïsation (Miller, 1995; Hill *et al.*, 2016; Zhang *et al.*, 2020). WordNet est un lexique informatique qui encode les concepts sous forme d'ensembles de synonymes (en anglais, *synsets*). Par exemple, le concept de *voiture* est représenté par l'ensemble {voiture, auto, automobile}. Plus récemment, BabelNet (Navigli et Ponzetto, 2012), un dictionnaire encyclopédique multilingue a été utilisé dans le but de faire une désambiguïsation multilingue du sens des mots (Kartsaklis *et al.*, 2020).

Reconnaissance d'entité nommée. Une entité nommée est un nom propre désignant une personne, une entreprise ou un lieu. Étant donné un flux de texte, on veut déterminer quels éléments du texte correspondent à des entités nommées et quel est leur type (personne, lieu, organisation, etc.). Bien que les lettres majuscules puissent aider à reconnaître les entités nommées dans des langues telles que l'anglais, ces informations ne suffisent pas à déterminer le type d'entité nommée et, dans tous les cas, sont souvent inexactes ou insuffisantes. Par exemple, la première lettre d'une phrase est également en majuscule et les entités nommées s'étendent souvent sur plusieurs mots, dont seuls certains sont en majuscule (Tjong Kim Sang et De Meulder, 2003). Cependant, d'autres langues, comme le chinois ou l'arabe, n'ont pas de notion de casse (Niu *et al.*, 2007).

Extraction terminologique. Le but de l'extraction terminologique est d'extraire automatiquement les termes pertinents d'un corpus donné (Alrehamy et Walker, 2017).

Liaison d'entités. De nombreux mots, généralement des noms propres, font référence à des entités nommées. On doit sélectionner l'entité, comme un individu célèbre, un lieu ou une entreprise, auxquels on fait référence dans le contexte (Hachey *et al.*, 2013).

1.1.5 Sémantique relationnelle

La sémantique relationnelle est la sémantique des phrases individuelles.

Étiquetage sémantique des rôles. Étant donné une phrase, on veut identifier et désambiguïser les prédicats sémantiques, comme les cadres verbaux, puis identifier et classer les éléments du cadre selon ses rôles sémantiques (Palmer *et al.*, 2010).

Analyse sémantique. Étant donné une phrase, on veut représenter formellement sa sémantique, soit sous forme de graphe, soit conformément à un formalisme logique (Shaw *et al.*, 2013; Zhang, 2020; Maulud *et al.*, 2021). Le défi englobe généralement des aspects connexes à plusieurs tâches plus élémentaires de la sémantique, par exemple l'étiquetage des rôles sémantiques et la désambiguïsation du sens des mots.

1.1.6 Analyse du discours

Il est aussi pertinent de représenter la sémantique du discours, c'est-à-dire la sémantique au-delà des phrases individuelles.

Analyse des sentiments. Extraire des informations subjectives à partir d'un ensemble de documents, en utilisant généralement les avis des utilisateurs dans le but de déterminer le sentiment général à l'égard d'objets spécifiques. Cela est particulièrement utile afin d'identifier les tendances de l'opinion publique dans les médias sociaux ou à des fins de publicité et de mise en marché (Maulud *et al.*, 2021).

Extraction de relations. Étant donné un morceau de texte, on désire identifier les relations entre les entités nommées, par exemple, qui est marié à qui, qui est le collègue de qui, etc. (Tian *et al.*, 2020).

Résolution de coréférence. À partir d'un morceau de texte, on doit déterminer quels mots font référence aux mêmes objets. La résolution d'anaphores est un exemple spécifique de cette tâche et concerne spécifiquement la mise en correspondance des pronoms avec les noms ou les noms auxquels ils se réfèrent (Bengtson et Roth, 2008).

Implication textuelle. Étant donné deux fragments de texte, on doit déterminer si l'un étant vrai implique l'autre, implique la négation de l'autre ou permet à l'autre d'être vrai ou faux (Bentivogli *et al.*, 2011).

Segmentation et reconnaissance des sujets. Étant donné un morceau de texte, on doit le séparer en segments dont chacun est consacré à un sujet. Ensuite on identifie le sujet du segment (Reynar, 1998).

1.1.7 Applications de niveau supérieur

Les applications de niveau supérieur s'attardent à des problèmes qui combinent plusieurs tâches plus simples et nécessitent une modélisation de plusieurs types de connaissances humaines, telles la grammaire, la sémantique et la connaissance factuelle sur le monde réel (Vauquois, 2003).

Traduction automatique. La demande de traduction s'est accrue ces dernières années en raison de l'augmentation de l'échange d'informations entre différentes régions utilisant différentes langues régionales (Okpor, 2014). La tâche consiste donc à traduire automatiquement un texte d'une langue humaine à une autre.

Génération de langage naturel. On cherche à convertir les informations des bases de données informatiques ou des propos sémantiques en langage humain lisible. Le premier livre généré par une machine a été créé en 1984 par Chamberlain à l'aide d'un système basé sur des règles (Styne, 1986).

Un chatbot est une application logicielle utilisée pour mener une conversation de chat en ligne via texte, au lieu de fournir un contact direct avec un humain. Les chatbots sont utilisés à diverses fins, notamment le service client, le routage des demandes et la collecte d'informations (Adamopoulou et Moussiades, 2020).

Résumé automatique de texte. Le but est de produire un résumé lisible d'un morceau de texte (Maulud *et al.*, 2021). Cette tâche est utilisée pour fournir des résumés de texte d'un type connu, tels que des documents de recherche et des articles dans la section financière d'un journal (Radford *et al.*, 2018).

Gestion des dialogues. En psychologie, l'attention conjointe consiste en un comportement dans lequel deux personnes se concentrent sur un objet ou un événement, dans le but d'interagir l'une avec l'autre (Kozima et Ito, 1997). Dans cet ordre d'idée, les systèmes informatiques destinés à converser avec un humain ont le potentiel d'améliorer les environnements de travail et la vie quotidienne de la population en général (Adamopoulou et Moussiades, 2020; Wollny *et al.*, 2021).

Correction d'erreurs grammaticales. La détection et la correction des erreurs grammaticales impliquent des problèmes à tous les niveaux de l'analyse linguistique dont la phonologie, l'ortho-

graphe, la morphologie, la syntaxe et la sémantique.

Réponse aux questions. Étant donné une question en langage humain, on doit déterminer sa réponse (Hirschman et Gaizauskas, 2001; Adamopoulou et Moussiades, 2020; Maulud *et al.*, 2021). Les questions typiques ont une bonne réponse spécifique, par exemple : « Quelle est la capitale du Canada ? ». Cependant, des questions ouvertes sont parfois également envisagées, par exemple : « Quel est le sens de la vie ? ».

1.2 Jeu du dictionnaire

Le jeu du dictionnaire [11] est un jeu de type sérieux visant à mieux comprendre le lexique mental. On commence avec un mot racine donné que le joueur doit définir. Les nouveaux mots contenus dans la définition sont catégorisés *à définir* et doivent donc être définis à leur tour. Le jeu s'arrête lorsqu'il n'y a plus de mots à définir. En d'autres termes, le joueur doit définir récursivement chacun des nouveaux mots contenus dans les définitions, jusqu'à ce que tous les mots soient définis. Notons qu'il y a un traitement qui s'effectue sur les définitions originales du joueur, c'est-à-dire qu'on effectue une lemmatisation de la définition dont les mots fonctionnels sont supprimés.

Lorsque le jeu se termine, alors on peut visualiser le dictionnaire du joueur sous forme de graphe lexical. Étant donné que tous les mots auront été définis, le graphe aura la propriété que, pour chaque mot représenté par un sommet, il y aura un chemin menant au mot racine.

À chaque étape, le joueur devra choisir entre les actions suivantes :

— Proposer une nouvelle définition : le joueur définit un mot. Pour chacun des nouveaux mots de la définition, s'il existe un chemin vers le mot racine, alors on dit d'eux qu'ils sont *actifs*. Sinon, ils sont *inactifs*. C'est donc dire qu'un mot *à définir* n'est pas défini, mais est *actif*.

— Supprimer une définition : le joueur peut supprimer une définition. Cela déplace dans la catégorie *à définir* le mot associé à la définition supprimée. Cela peut également mener au cas où le graphe se déconnecte, c'est-à-dire que pour certains mots, il n'y a plus de chemin menant au mot racine. On doit alors vérifier récursivement tous les mots du dictionnaire afin de différencier les mots *actifs* des mots *inactifs*.

11. https ://combinat.info.uqam.ca

— Mettre à jour une définition : cela est équivalent à supprimer une définition, puis à produire une nouvelle définition.

Plus formellement, un graphe orienté ou digraphe est un graphe dans lequel les arêtes ont des orientations (Bang-Jensen et Gutin, 2008). Un digraphe est un couple ordonné $G = (V, E)$ comprenant :

— V, un ensemble de sommets

— E, un ensemble d'arêtes dirigées ou arcs qui sont des paires de sommets, c'est-à-dire qu'une arête est associée à deux sommets distincts. Dans l'arc (u, v), u désigne la source de l'arc et v désigne la cible de l'arc. On dit que l'arc est incident aux sommets u et v, c'est-à-dire que ces sommets sont l'une des extrémités de l'arc. On appelle arêtes multiples deux arêtes ou plus avec la même source et la même cible. Les arêtes multiples ne sont pas permises dans un digraphe simple.

Plus généralement, on considère V et E comme étant des ensembles finis. De plus, V est souvent supposé non vide, mais E est autorisé à l'être. Un sommet peut exister dans un graphe et ne pas avoir d'arcs incidents. L'ordre d'un graphe, noté $|V|$, désigne son nombre de sommets. La taille d'un graphe, notée $|E|$, désigne son nombre d'arêtes. Le degré d'un sommet est le nombre d'arêtes qui lui sont incidentes, où une boucle est comptée deux fois. Le degré d'un graphe est le maximum des degrés de ses sommets.

Un parcours (en anglais, *walk*) est une séquence finie ou infinie d'arêtes qui rejoint une séquence de sommets. Un sentier (en anglais, *trail*) est un parcours dans lequel toutes les arêtes sont distinctes. Un chemin (en anglais, *path*) est un sentier dans lequel tous les sommets sont distincts. Un circuit dirigé est un chemin dirigé non vide dans lequel les premier et dernier sommets sont les mêmes.

Un digraphe est dit faiblement connexe si le remplacement de toutes ses arêtes dirigées par des arêtes non orientées produit un graphe non orienté connexe. Il est semi-connexe s'il contient un chemin de u vers v ou un chemin de v vers u pour chaque paire de sommets (u, v). Il est fortement connexe, s'il contient un chemin de u vers v et aussi un chemin de v vers u pour chaque paire de sommets (u, v). Les composantes fortement connexes d'un digraphe sont les sous-graphes maximaux fortement connexes.

Par définition, le jeu du dictionnaire résulte en un digraphe simple, enraciné et faiblement connexe.

La figure 1.2 illustre une partie terminée du jeu dont le mot racine *cheval* est représenté en caractère gras.

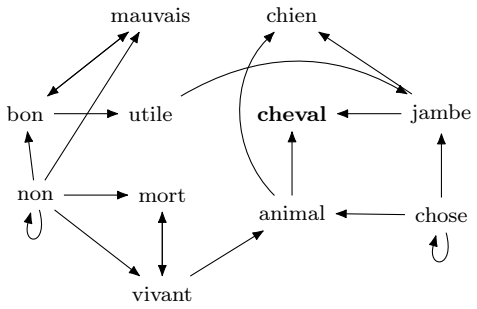

FIGURE 1.2 – Le mot racine **cheval** est identifié en caractère gras. Les définitions du joueur passent par un filtre des mots fonctionnels puis par une lemmatisation. Par exemple, le mot *cheval* a été défini par le joueur avec : $P_{\text{cheval}} = \langle$animal, avec, des, jambes$\rangle$. Après le traitement, la définition devient : $P_{\text{cheval}} = \langle$animal, jambe$\rangle$. Lorsque le jeu se termine, le dictionnaire peut alors être représenté par un graphe orienté, où les sommets sont des symboles définis par les sommets qui leur sont connexes. Le graphe a aussi la propriété que, pour chaque sommet, il existe un chemin vers le mot racine.

Bien que l'on puisse apprendre de nouvelles significations de mots à partir d'un dictionnaire, un dictionnaire entier ne peut pas être appris seulement par définition en raison des références circulaires dans les définitions. Dans des recherches antérieures, Blondin Massé et al. extraient des dictionnaires leur noyau fondamental, à partir duquel le reste du dictionnaire peut être atteint par les seules définitions (Blondin Massé *et al.*, 2008; Chicoisne *et al.*, 2008). Ce noyau présente des propriétés qui se reflètent également dans le lexique mental, puisqu'ils constatent, notamment, que les mots du noyau de base sont plus fréquents à l'oral et à l'écrit, plus concrets, plus facilement imagés et appris à un plus jeune âge (Chicoisne *et al.*, 2008). Ainsi, on cherche à savoir dans quelle mesure un dictionnaire est sémantiquement développé. Puisque le langage est la capacité d'exprimer une pensée et de communiquer, nous croyons que la notion de dictionnaire sémantiquement développé peut être vu comme étant un dictionnaire qui réduit l'ambigüité autour de certain cas de figure sémantique pour un groupe d'individu donnée.

Le tableau 1.1 contient 3 *definienda*[12] pour un *définiens*[13] illustrant certain cas de figure d'ordre sémantique :

12. [...] unité lexicale à définir [...] (Riegel, 1987)

13. [...] une séquence définissante [...] (Riegel, 1987)

Cas	Definiens	Definienda
spécifique	fruit du pommier	pomme, McIntosh, Cortland
antonymie	contraire du contraire	pareil, identique, analogue
hyponymie	véhicule transportant des passagers	autobus, train, avion
hyperonymie	lion ou tigre ou chat	animal, panthera, félin
ambigüe	un fruit rouge	pomme, cerise, fraise
polysémie	c'est de la tarte	dessert, facile, sucre
vague	une chose vivante	animal, humain, arbre
ordre	entre 9pm et 6am	nuit, soir, minuit
ordre	entre 6am et 9pm	jour, matin, midi

TABLE 1.1 – Pour diminuer l'ambiguïté autour du sens d'une définition, un groupe d'individu doit communiquer dans un langage qui permet de capturer les relations de polysémies, d'antonymies, d'hyponymies parmi d'autres relations sémantiques entre les mots.

À partir de cela, nous émettons l'hypothèse qu'en conjecturant un mot à partir de sa définition, on sera en mesure d'établir une mesure qualitative d'un dictionnaire.

1.3 Notation mathématique

Cette section établit la notation mathématique qui est utilisée dans le mémoire dans le cadre du problème du dictionnaire inversé.

Nous désignons le fait que le symbole w fasse partie de la phrase P par la notation $w \in P$ même si P n'est techniquement pas un ensemble. Sous cette classification, un dictionnaire D peut être défini comme un ensemble de paires (w, P), où P désigne la définition du mot w.

La fonction d'association d'un symbole w d'un dictionnaire, notée $m(w)$, retourne l'ensemble des définitions $\{P_1, P_2, \ldots, P_n\}$ associé à w, c'est-à-dire $m(w) = \{P \mid (w, P) \in D\}$. Par exemple, considérons que, dans un dictionnaire, le mot *pomme* soit uniquement défini par les paires suivantes :
(**pomme**, ⟨Fruit, du, pommier, charnu, de, forme, plus, ou, moins, arrondie, de
couleur, verte, jaune, ou, rouge, selon, la, variété, que, l', on, consomme, frais, en,
compote, en, beignets, et, dont, on, fait, le, cidre, ou, des, jus⟩),
(**pomme**, ⟨Objet, partie, d', objet, ou, ornement, de, forme, arrondie⟩).

Dans ce cas, $m(pomme)$ retournerait un ensemble de deux éléments. Notons que la fonction d'association inverse de w, notée $m^{-1}(w)$ retourne l'ensemble de tous les symboles contenant w dans leurs définitions.

Pour le problème du dictionnaire inversé, l'objectif est de construire un modèle M, paramétré par Θ, retournant un ensemble de candidats C, pour lequel il existe $w \in C$ tel que $P \in m(w)$, où $M(P)$ minimise une fonction de coût J quantifiant la précision des prévisions du modèle. Autrement dit, le modèle doit maximiser la probabilité $p(w \mid P; D)$, c'est-à-dire la probabilité qu'un mot w soit de pair avec P dans D. Généralement, afin de limiter le nombre de candidats, on fixe un entier k et on ne retient que les k meilleurs candidats selon la fonction de coût J.

Afin de réduire l'ambiguïté autour de la notation des variables dans ce mémoire, celles-ci ont, sauf mention contraire, été uniformisées. Pour consulter la notation, se référer à la fiche de notation située dans le glossaire.

CHAPITRE 2
MÉTHODE STATISTIQUES ET VECTORIELLES

Dans ce chapitre, on s'intéresse à la résolution du problème du dictionnaire inversé à l'aide de méthodes statistiques comprenant les arbres d'analyse syntaxique, les matrices de similarité et la méthode hongroise. On détaille aussi l'encodage des mots avec les méthodes de sac de mots, les matrices de cooccurrences, la méthode statistique TF-IDF et les plongements vectoriels, en particulier Word2vec.

2.1 Résolution à l'aide d'une base de données

L'approche consistant à comparer la phrase P à chaque définition d'un dictionnaire présente un problème majeur : dans un dictionnaire contenant plus de 100 000 mots définis, dans lequel chaque mot peut avoir plusieurs définitions, il faut calculer la similarité entre la phrase P et chacune des définitions du dictionnaire des centaines de milliers de fois avant d'identifier les meilleurs mots candidats. Il est possible d'améliorer l'algorithme, notamment en utilisant la fonction d'association inverse $m^{-1}(w)$ qui retourne l'ensemble de toutes les définitions dans lesquelles w apparaît. Afin de résoudre le problème du dictionnaire inversé, Shaw et autres proposent un algorithme dans lequel on réduit l'espace de recherche puisqu'on sait dans quelles définitions un mot donné apparaît (Shaw *et al.*, 2013).

2.1.1 Prétraitement

On supprime d'abord tous les mots fonctionnels de P. On construit une requête Q en reliant avec le symbole ET les symboles de la phrase P. On considère ensuite les cas où il est possible de modifier par une négation un symbole contenu dans P. Par exemple, les mots *pas agréable* sont sémantiquement similaires à *désagréable* ou *incommodant*. Ensuite, on modifie Q afin d'inclure les antonymes, en les concaténant avec le symbole OU. Par exemple, $Q = (\textit{humidité ET pas ET agréable})\ OU\ (\textit{humidité ET}\ (\textit{déplaisant OU malheureux}))$.

On extrait la racine de chaque mot de Q, puis on normalise Q de telle sorte que tous les noms

apparaissent en premier, puis les verbes, puis les adjectifs et les adverbes. Les auteurs de l'algorithme affirment que les noms et les verbes représentent des concepts plus concrets que les adjectifs ou les adverbes, puisque selon la fonction d'association m^{-1}, ils retournent, en moyenne, des ensembles plus petits que les adjectifs et les adverbes (Shaw *et al.*, 2013).

On obtient l'ensemble O des phrases à considérer en évaluant Q en tenant compte de la fonction inverse, c'est-à-dire que $O = \{m^{-1}(w) \mid w \in Q\}$. L'ensemble O doit être de taille α, où α est un hyperparamètre. Si $O \geq \alpha$, alors on trie les résultats selon une mesure de similarité décrite plus bas. Sinon, on modifie Q afin de prendre en compte les synonymes, les hyponymes et les hyperonymes, puis on réévalue Q selon m^{-1}. Si la taille de O n'est toujours pas, au minimum, de taille α, on généralise Q en supprimant successivement un symbole, par ordre décroissant d'occurrence dans O. Cela supprime d'abord les mots les plus courants, en partant du principe que les mots les moins courants seront les plus importants afin de trouver les bons candidats. On boucle ainsi jusqu'à ce qu'on obtienne une sortie suffisante, ou jusqu'à ce que Q ne contienne que deux symboles.

2.1.2 Matrice de similarité

Pour mesurer l'importance d'un symbole w_1 par rapport au symbole w_2, Shaw et al. utilisent l'analyseur de phrase OpenNLP, qui retourne la structure grammaticale d'une phrase et qu'on note $\mathrm{T}(P)$ (Foundation, 2009). On considère $w_1 \in \mathrm{T}(P')$ où $P' \in O$ et $w_2 \in \mathrm{T}(P)$. On calcule d'abord la similarité $\mathrm{sim}(w_1, w_2)$ en fonction de leur emplacement dans la hiérarchie de WordNet. Cette hiérarchie organise les mots du plus général à la racine vers le plus spécifique dans les feuilles. Intuitivement, deux symboles ont très peu de similitudes si leur plus petit ancêtre commun (LCA pour *least common ancestor*) dans cette hiérarchie est la racine. La fonction $\mathrm{LCA}(w_1, w_2)$ retourne le LCA partagé par w_1 et w_2 dans la hiérarchie. La fonction $d(w)$ retourne la profondeur d'un symbole w dans la hiérarchie. La fonction $\mathrm{sim}(w_1, w_2)$ croît lorsque w_1 et w_2 sont proches de leur LCA.

On évalue ensuite la notion d'importance d'un symbole, noté $\lambda(w, P)$, pour un mot w dans une phrase P. Par exemple, considérons les phrases suivantes :

$$\begin{aligned} P_1 &= \langle \text{le, chien, qui, a, mordu, l', homme} \rangle , \\ P_2 &= \langle \text{l', homme, qui, a, mordu, le, chien} \rangle , \\ P_3 &= \langle \text{le, chien , agité, qui, a, mordu, l', homme} \rangle . \end{aligned} \tag{2.1}$$

Le sujet de la phrase est intuitivement plus important que l'objet. Pour P_1, *chien* est plus important que *homme*, et pour P_2, *homme* est plus important que *chien*. Shaw et al. affirment que si on devait ajouter des adjectifs ou des adverbes, ceux-ci ajouteraient des détails, mais seraient négligeables pour le sens de la phrase. Par exemple, P_3 diffère peu de P_1 d'un point de vue sémantique. Alors, pour générer une mesure de l'importance d'un mot dans une phrase, Shaw et al. pondèrent la profondeur d'un mot, notée $d(w)$, sur la profondeur totale de l'arbre, notée $d(T(P))$. Le facteur $\lambda(w, P)$ croît davantage lorsque w est situé haut dans l'arbre d'analyse.

Cela se traduit par les équations :

$$\begin{aligned} \mathrm{sim}(w_1, w_2) &= \frac{2 \times d(\mathrm{LCA}(w_1, w_2))}{d(w_1) + d(w_2)}, \\ \lambda(w, P) &= 1 - \frac{d(w)}{d(\mathrm{T}(P))}. \end{aligned} \tag{2.2}$$

Puis, un facteur de similarité entre deux mots de deux phrases suit l'équation :

$$\mu(w_1, P, w_2, P') = \lambda(w_1, P) \times \lambda(w_2, P') \times \mathrm{sim}(w_1, w_2), \tag{2.3}$$

où

— $w_1 \in P$,

— $w_2 \in P'$ avec $P' \in O$.

Ensuite, on calcule une matrice selon le facteur de similarité entre tous les mots de la phrase d'entrée et tous les mots d'une phrase provenant de l'ensemble O. Cette matrice est utilisée comme entrée à la méthode hongroise.

2.1.3 Méthode hongroise

La méthode hongroise est une technique d'optimisation qui résout le problème d'affectation du travail (en anglais, *job assignment problem*) en temps polynomial (Kuhn, 1955). Le problème d'affectation du travail cherche à attribuer chaque travail à exactement un opérateur, de sorte que la durée totale (coût) des affectations soit minimum (Kauser, 2016). La méthode hongroise prend en entrée une

matrice équilibrée. Si la matrice n'est pas équilibrée, on doit ajouter des colonnes et/ou des lignes dont les cellules auront une valeur initialisée à zéro. La méthode comprend 4 étapes :

1. Soustraire les valeurs minimales de chaque ligne de la matrice des entrées de cette ligne
2. Soustraire les valeurs minimales de chaque colonne de la matrice des entrées de cette colonne
3. Tracer un minimum de lignes pour couvrir tous les zéros de la matrice avec la procédure suivante :
   (a) parcourir les lignes

   Pour chacune des lignes, en commençant par la première, s'il y a exactement un zéro non couvert dans cette ligne, marquer d'un cercle cette entrée zéro et tracer un trait vertical dans la colonne de cette entrée zéro. Sinon, sauter cette ligne. Après avoir scanné la dernière ligne, vérifier que tous les zéros sont couverts par les traits. Si oui, passer à l'étape 4, sinon parcourir les colonnes.

   (b) parcourir les colonnes

   Pour chacune des colonnes, en commençant par la première, s'il y a exactement un zéro non couvert dans cette colonne, marquer d'un cercle cette entrée zéro et tracer un trait horizontal dans la ligne de cette entrée zéro ; Sinon, sauter cette colonne.
4. Si le nombre de cercles est égal au nombre de lignes de la matrice, retourner la somme des valeurs de la matrice d'entrée correspondant aux indices des cellules encerclées. Sinon on identifie la valeur minimale des valeurs de cellule non marquée par un trait. Ensuite, on procède de la façon suivante :
   (a) Ajouter la valeur minimale aux valeurs qui se trouvent au point d'intersection des traits
   (b) Soustraire la valeur minimale de toutes les valeurs de cellule non marquée par un trait
   (c) Toutes les autres entrées restent les mêmes
   (d) Revenir à l'étape 3

La figure 2.1 illustre les différentes étapes de l'algorithme de la méthode hongroise pour un cas où, à l'étape 4, on doit retourner à l'étape 3.

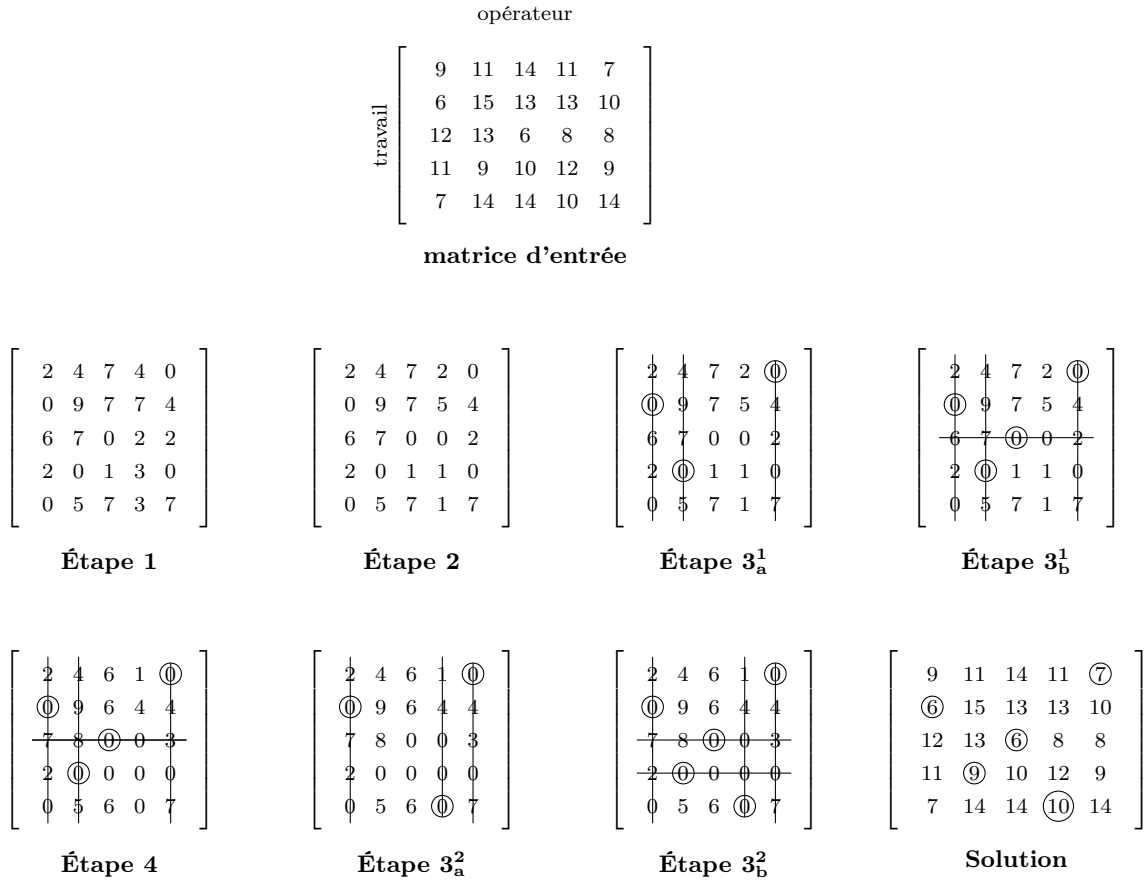

FIGURE 2.1 – On débute avec une matrice d'entrée équilibrée de taille $n \times n$. Une solution optimale survient lorsque chaque ligne et chaque colonne possède exactement une cellule encerclée dont la valeur est égale à zéro. On retourne alors la somme des valeurs qui se retrouvent aux indices associés aux cellules encerclées dans la matrice d'entrée. Dans le cas présent, on trouve une solution au coût minimal de $7 + 6 + 6 + 9 + 10 = 38$.

Cette méthode peut être utilisée afin de trouver la similarité entre deux phrases en inversant l'objectif, c'est-à-dire qu'au lieu de minimiser le coût, on cherche à maximiser les relations sémantiques entre les mots de deux phrases. On utilise donc la matrice de similarité obtenue en 2.1.2 comme entrée à la méthode hongroise afin d'obtenir une mesure de similarité entre les phrases de l'ensemble O et la phrase d'entrée P (Kuhn, 1955; Gao *et al.*, 2015).

Base de données. Pour effectuer le traitement, on a besoin d'accéder aux informations enregistrées dans des bases de données pour chaque mot :

1. la base de données des définitions originales du dictionnaire.

2. la base de données des arbres d'analyse.
3. la base de données de synonymes
4. la base de données des hyponymes et hyperonymes
5. la base de données d'antonymes

La figure 2.2 illustre comment le modèle utilise des requêtes sur les bases de données afin de retourner un résultat pour le problème du dictionnaire inversé.

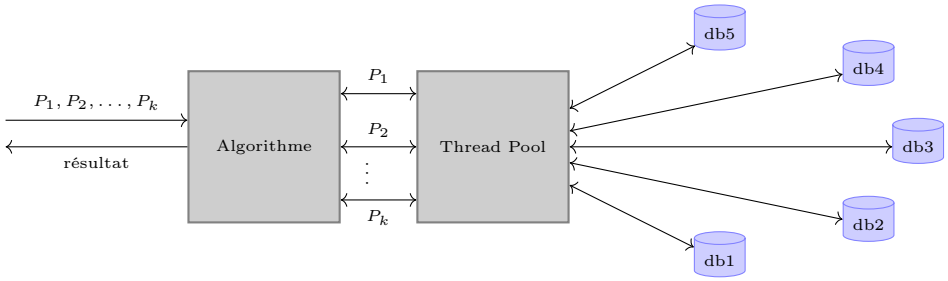

FIGURE 2.2 – L'algorithme traite chacune des phrases P en faisant des requêtes sur les différentes bases de données pour chacun des mots d'une phrase P.

2.2 Représentation vectorielle

Dans le domaine du traitement automatique du langage, l'expression *plongements de mots* est un terme utilisé pour dénoter la représentation de mots, généralement sous la forme d'un vecteur à valeur réelle qui encode la signification du mot de telle sorte que les mots qui sont plus proches sémantiquement aient une signification similaire (Daniel et James, 2000). Les plongements de mots peuvent être obtenus à partir d'une modélisation du langage dans laquelle des mots du vocabulaire sont associés à des vecteurs de nombres réels. Les méthodes pour générer les plongements utilisent notamment des représentations explicites, des modèles probabilistes, des matrices de cooccurrence et des réseaux de neurones (Heidenreich, 2018).

2.2.1 Encodage *one-hot*

Étant la plus simple de toutes les techniques, un encodage *one-hot* utilise un vecteur épars afin de représenter chaque mot du vocabulaire. Pour chacun des mots, cela se traduit par un vecteur v_w, de dimension $|V|$, initialisé à zéro sauf pour l'indice du bit *hot* associé à l'indice du mot dans le dictionnaire. Par exemple, prenons un dictionnaire constitué uniquement de trois mots, lesquels

seraient respectivement encodés *one-hot* de cette manière : $v_1 = [100], v_2 = [010], v_3 = [001]$. Un inconvénient de cet encodage est qu'il ne capture pas de signification sémantique profonde pour un mot par rapport à un autre, puisque tous les vecteurs sont équidistants les uns des autres (Srilatha *et al.*, 2021).

Similarité. Idéalement, on souhaite que les mots similaires comme *roi* et *reine* aient des caractéristiques similaires. Mais avec l'encodage *one-hot*, le vecteur pour le mot *roi* devient aussi similaire au vecteur du mot *reine* autant qu'à n'importe quel autre mot suivant cet encodage. De surcroît, on voudrait faire des opérations vectorielles sur ces représentations de mots. Pour ce faire, on a besoin d'une représentation suffisamment riche afin d'avoir des identités telles que : $v_{\text{roi}} - v_{\text{homme}} + v_{\text{femme}} \approx v_{\text{reine}}$ (Pennington *et al.*, 2014).

Taille du vocabulaire. Avec cette approche, la taille des vecteurs de représentation augmente proportionnellement à la taille du vocabulaire. Il y a des raisons pour lesquelles on ne veut pas que la taille des caractéristiques explose, puisque cela amène vers la malédiction des dimensions. En règle générale, pour bien généraliser, un modèle est formé à partir de plus de données d'entraînement qu'il n'y a de fonctionnalités (Köppen, 2000).

Coût computationnel. Le vecteur de représentation serait principalement composé de zéros et les réseaux de neurones en particulier fonctionnent moins bien avec des vecteurs épars (Yang et Ma, 2019). Avec un si grand espace de fonctionnalités, on risque également de rencontrer des problèmes de mémoire (Yin *et al.*, 2012).

2.2.2 Score TF-IDF

Le score TF-IDF désigne un rapport entre la fréquence d'un mot (en anglais, TF pour *terme-frequency*) et de la fréquence inverse de document (en anglais, IDF pour *inverse document frequency*). Le score TF-IDF pour un mot dans un document d est calculé en prenant leur produit. Plus le score est élevé, plus ce mot est pertinent dans un document particulier (Achananuparp *et al.*, 2008).

Fréquence des termes. Il existe plusieurs façons de calculer la fréquence d'un terme (mot) dans un document. Une façon simple consiste à dénombrer l'occurrence du mot dans un document, puis

de pondérer le résultat soit, par exemple, par calcul brut (divisé par 1), par fréquence booléenne (1 si $w \in d$, sinon 0), par fréquence sur une échelle logarithmique $1 + \log(\text{freq}(w, d))$ ou par fréquence relative.

Utiliser la fréquence relative pour calculer la fréquence d'un mot w dans un document d se traduit par l'équation :

$$\text{tf}(w, d) = \frac{\text{freq}(w, d)}{\sum_{w' \in d} \text{freq}(w', d)}. \tag{2.4}$$

Fréquence inverse de document. La fréquence inverse de document signifie à quel point un mot est courant ou rare dans l'ensemble des documents D. Par exemple, puisque que le terme *le* est très courant, la fréquence des termes aura tendance à mettre l'accent sur les documents qui utilisent plus fréquemment ce terme. Or, le terme *le* est un mot fonctionnel sans grande valeur sémantique qui ajoute du bruit aux données. Celui-ci n'est donc pas un bon choix afin de distinguer les documents et termes pertinents, contrairement à des mots de contenu tels que *marron* ou *vache* (Wilbur et Sirotkin, 1992). Par conséquent, un facteur de fréquence inverse de document est ajouté, ce qui diminue le poids des termes qui apparaissent très fréquemment dans l'ensemble de documents et augmente le poids des termes qui apparaissent rarement. Cette métrique peut être calculée en prenant le logarithme du nombre total de documents sur le nombre de documents contenant un mot donné.

Cela se traduit par l'équation :

$$\text{idf}(w, D) = \log\left(\frac{|D|}{|d \in D : w \in d|}\right), \tag{2.5}$$

où

— $|d \in D : w \in d|$: nombre de documents dans lequel un mot w apparaît.

Si le mot est très courant et apparaît dans de nombreux documents, la fréquence inverse de document tendra vers 0. Autrement dit, les mots qui apparaissent souvent et un peu partout, par exemple *le* ou *et*, devraient avoir peu de poids ou de signification, puisqu'ils n'ont pas une grande valeur sémantique. Cependant, si un mot apparaît très peu ou fréquemment, mais à seulement un ou deux endroits, alors c'est probablement un mot plus important qui doit être pondéré comme tel.

On multiplie ensuite les deux équations précédentes pour obtenir le score TF-IDF :

$$\text{tf-idf}(w, d, D) = \text{tf}(w, d) \times \text{idf}(w, D). \tag{2.6}$$

Exemple.

On considère un corpus contenant deux documents où certains mots apparaissent selon une fréquence telle que mentionnée dans les tableaux ci-desous :

Document $d1$

mot	fréquence
le	1
ciel	1
est	2
bleu	1

Document $d2$

mot	fréquence
le	1
chien	1
est	2
calme	3

Alors, les scores TF-IDF pour le mot *le* se traduisent par les équations suivantes :

$$\begin{aligned}
&\text{tf}(le, d1) = \frac{1}{5} = 0.20, \\
&\text{tf}(le, d2) = \frac{1}{7} \approx 0.14, \\
&\text{idf}(le, D) = \log\left(\frac{2}{2}\right) = 0, \\
&\text{tf-idf}(le, d1, D) = 0.20 \times 0 = 0, \\
&\text{tf-idf}(le, d2, D) = 0.14 \times 0 = 0.
\end{aligned} \tag{2.7}$$

De même, les scores TF-IDF pour le mot *calme* se traduisent par les équations suivantes :

$$\begin{aligned}
&\text{tf}(calme, d1) = \frac{0}{5} = 0, \\
&\text{tf}(calme, d2) = \frac{3}{7} \approx 0.429, \\
&\text{idf}(calme, D) = \log\left(\frac{2}{1}\right) = 1, \\
&\text{tf-idf}(calme, d1, D) = 0 \times 1 = 0 \\
&\text{tf-idf}(calme, d2, D) = 0.429 \times 1 = 0.429.
\end{aligned} \tag{2.8}$$

Afin de plonger un mot dans un espace vectoriel avec un score TF-IDF, on utilise la même méthode que le *one hot* en remplaçant le bit *hot* par la valeur TF-IDF. Ainsi, L'utilisation de cette technique afin de plonger les mots dans un espace vectoriel souffre également du problème des représentations épars qui ne capture pas la relation sémantique entre les mots (Srilatha *et al.*, 2021).

2.2.3 Matrice de cooccurrence

Une matrice de cooccurrence est une matrice de taille $|V| \times |V|$ qui se résume à une représentation numérique indiquant pour chaque paire de symboles combien de fois ils apparaissent ensemble dans une taille de fenêtre donnée. Cette méthode souffre également de lacunes quant aux relations sémantiques, à l'éparsité vectorielle et en coût computationnel (Srilatha *et al.*, 2021).

2.2.4 Sac de mots

Un sac de mots (BoW pour *bag-of-words*) est une modélisation représentant l'occurrence de mots dans un document. Cela implique d'avoir un vocabulaire de mots connus et une mesure de quantification des mots (Kandola *et al.*, 2002; Goldberg, 2017).

Parmi les méthodes de quantification des mots, on s'intéresse à trois variantes :

1. **Binaire** : Présence ou absence des mots.
2. **Cardinalité** : Le nombre de fois où chaque mot apparaît.
3. **Fréquence** : La fréquence à laquelle chaque mot apparaît parmi tous les mots d'un corpus.

Le modèle ne s'intéresse qu'à savoir si des mots connus apparaissent ou pas. On appelle cela un sac de mots, car on ignore l'information sur l'ordre ou la structure des mots.

Exemple.

Prenons le corpus « Une pomme est un fruit rouge. Un homard est un crustacé rouge ou bleu » avec lequel on obtient l'ensemble V :

$$\begin{aligned} &V = \{\text{une, pomme, est, un, fruit, rouge, homard, crustacé, ou, bleu}\}, \\ &|V| = 10. \end{aligned} \tag{2.9}$$

Soit P_1, P_2, P_3 trois suites de symboles tirés du corpus :

$$\begin{aligned} P_1 &= \langle \text{Un, fruit, rouge} \rangle , \\ P_2 &= \langle \text{Un, crustacé, rouge} \rangle , \\ P_3 &= \langle \text{Une, pomme, rouge} \rangle . \end{aligned} \tag{2.10}$$

En prenant la quantification binaire d'un sac de mots, on obtient trois vecteurs associés aux suites de symboles, soit v_1, v_2 et v_3 :

$$\begin{aligned} v_1 &= \text{BoW}(P_1) = [0,0,0,1,1,1,0,0,0,0], \\ v_2 &= \text{BoW}(P_2) = [0,0,0,1,0,1,0,1,0,0], \\ v_3 &= \text{BoW}(P_3) = [1,1,0,0,0,1,0,0,0,0]. \end{aligned} \tag{2.11}$$

Distance cosinus.

La similarité cosinus est utilisée pour déterminer la similarité entre des vecteurs. Elle mesure simplement le cosinus de l'angle entre deux vecteurs projetés dans un espace multidimensionnel. L'angle permet de trouver la distance entre deux vecteurs puisque dans les notions de géométrie euclidienne, les points de l'espace sont définis en fonction de leurs coordonnées cartésiennes. Dans une telle représentation, les notions de longueur et d'angles sont définies au moyen du produit scalaire (Strang *et al.*, 1993).

Exemple.

En se basant sur l'exemple précédent, on calcule les produits scalaires associés à v_1 :

$$\begin{aligned} v_1 \cdot v_1 &= \sum_i^{|V|} v_{1_i} v_{1_i} = 0+0+0+1+1+1+0+0+0+0 = 3, \\ v_1 \cdot v_2 &= \sum_i^{|V|} v_{1_i} v_{2_i} = 0+0+0+1+0+1+0+0+0+0 = 2, \\ v_1 \cdot v_2 &= \sum_i^{|V|} v_{1_i} v_{3_i} = 0+0+0+0+0+1+0+0+0+0 = 1. \end{aligned} \tag{2.12}$$

La k norme est dénotée par :

$$\|v\|_k = \left(\sum_{i=1}^{n} |v_i|^k \right)^{1/k}. \tag{2.13}$$

Ainsi

$$\begin{aligned} \|v_1\|_2 &= 1.73, \\ \|v_2\|_2 &= 2.45, \\ \|v_3\|_2 &= 1.73. \end{aligned} \tag{2.14}$$

De sorte que :

$$\begin{aligned} \cos(\theta) &= \frac{v_1 \cdot v_2}{\|v_1\|_2 \cdot \|v_2\|_2}, \\ J &= 1 - \cos(\theta), \\ J_{v_1 v_1} &= 1 - \frac{3}{1.73 \cdot 1.73} = 0, \\ J_{v_1 v_2} &= 1 - \frac{3}{1.73 \cdot 2.45} = 0.29, \\ J_{v_1 v_3} &= 1 - \frac{1}{1.73 \cdot 1.73} = 0.66. \end{aligned} \tag{2.15}$$

Par rapport à P_1, on obtient une distance cosinus de 0, 0.29 et 0.66 comparé à P_1, P_2 et P_3 respectivement.

Une approche simple consiste à comparer la distance cosinus entre chaque définition d'un dictionnaire et une phrase donnée P. On choisit ensuite les k définitions les plus proches de P.

Cette approche souffre de quelques lacunes :

1. **Vocabulaire** : Le vocabulaire nécessite une modélisation adéquate, notamment pour gérer la taille des vecteurs, ce qui impacte la qualité des représentations (Srilatha *et al.*, 2021).
2. **Éparsité** : Les représentations éparses sont plus difficiles à modéliser à la fois pour des raisons de calcul et d'information, où, pour les modèles, le défi consiste à exploiter peu d'informations dans un grand espace de représentation (Yang et Ma, 2019).
3. **Signification** : Cette stratégie ignore le contexte et la sémantique des mots (Mikolov *et al.*, 2013a).

2.2.5 Plongement de mots

Le but d'un plongement de mots (en anglais, *word embedding*) est de capturer des relations dans un espace vectoriel, que ce soit le sens, la morphologie, la polysémie ou un autre type de relation. Les plongements de mots encodent les mots numériquement dans un espace vectoriel, tout en réduisant significativement le nombre de dimensions nécessaires à l'encodage, comparativement aux autres méthodes énoncées précédemment (Zamani et Croft, 2016). Étant donné la relation sémantique des mots dans leur plongement, des opérations arithmétiques peuvent être faites sur ceux-ci (Pennington *et al.*, 2014).

La figure 2.3 illustre un tel exemple, et dans lequel la relation $v_{\text{roi}} - v_{\text{homme}} + v_{\text{femme}} \approx v_{\text{reine}}$ est satisfaite.

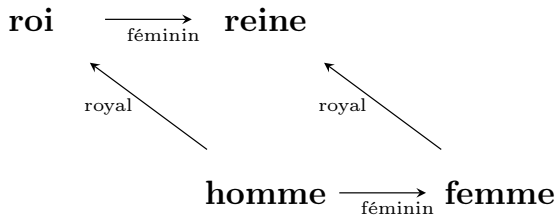

FIGURE 2.3 – Adapté de (Ethayarajh *et al.*, 2019). Au sens le plus large, une analogie de mots est une déclaration de la forme « a est à b comme x est à y », qui affirme que a et x peuvent être transformés de la même manière pour obtenir b et y, et vice-versa. Plus formellement, une analogie de mots est une transformation inversible qui s'applique à un ensemble de paires de mots ordonnés si et seulement si, pour tout $(x, y) \in S, f(x) = y$ et $f^{-1}(y) = x$. Lorsqu'il s'agit d'analogies linéaires de mots, elles forment un parallélogramme dans l'espace du plongement (Ethayarajh, 2019). Selon cette conjecture, l'analogie $v_{\text{roi}} - v_{\text{homme}} + v_{\text{femme}} \approx v_{\text{reine}}$ tient si et seulement si pour chaque mot w du vocabulaire on a $\frac{p(w|roi)}{p(w|reine)} \approx \frac{p(w|homme)}{p(w|femme)}$ (Pennington *et al.*, 2014).

Il a été démontré que les plongements de mots, lorsqu'ils sont utilisés comme représentation d'entrée d'un modèle, offrent de meilleures performances sur des tâches du TAL, telles que l'analyse syntaxique et l'analyse des sentiments (Socher *et al.*, 2013). Il n'existe pas de nombre théorique optimal de dimensions puisque cela s'avère être une question empirique qui dépend du problème. Augmenter le nombre de dimensions signifie une représentation des mots plus précise, mais signifie également une demande plus élevée de ressources de calcul. Un compromis doit aussi être fait entre la précision du modèle et le coût computationnel. En général, on voit des plongements de mots de dimensions variant entre 50 et 500 (Mikolov *et al.*, 2013a; Devlin *et al.*, 2019).

La figure 2.4 illustre une matrice de plongement pour un vocabulaire de jetons commençant par *a* et se terminant par *zythum*.

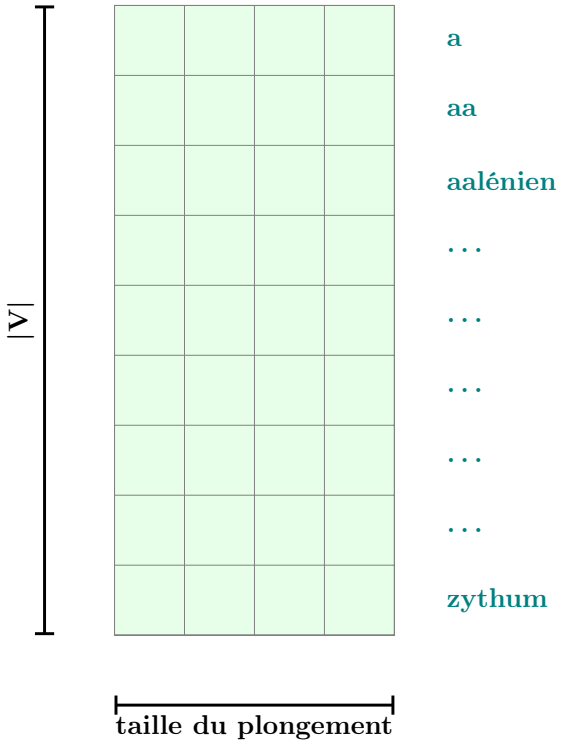

FIGURE 2.4 – Adapté de (Jalamar, 2020a). On associe à chaque mot un vecteur de dimensions de la taille du plongement dont les valeurs sont contenues entre [0,1] et où les mots sémantiquement similaires ont des vecteurs semblables. Le défi consiste souvent à capturer, en autres, la polysémie et la morphologie d'un langage. L'approche la plus courante utilise des réseaux de neurones afin de plonger les mots dans une représentation vectorielle contextuelle (Radford *et al.*, 2018; Devlin *et al.*, 2019; Pennington *et al.*, 2014; Mikolov *et al.*, 2013a).

2.2.5.1 Analyse en composantes principales

L'analyse en composantes principales consiste à calculer les composantes principales de données et à les utiliser afin d'effectuer un changement de base, en utilisant parfois que les premières composantes principales et en ignorant les autres (Wold *et al.*, 1987). Il est alors possible d'analyser des mots d'un vocabulaire dans des représentations vectorielles en haute dimension en réduisant les dimensions, c'est-à-dire en les projetant dans une représentation visuellement interprétable.

La figure 2.5 illustre l'emplacement de quelques mots dans un espace à trois dimensions après une analyse des composantes principales du transformeur GPT-2 (voir section 4.3).

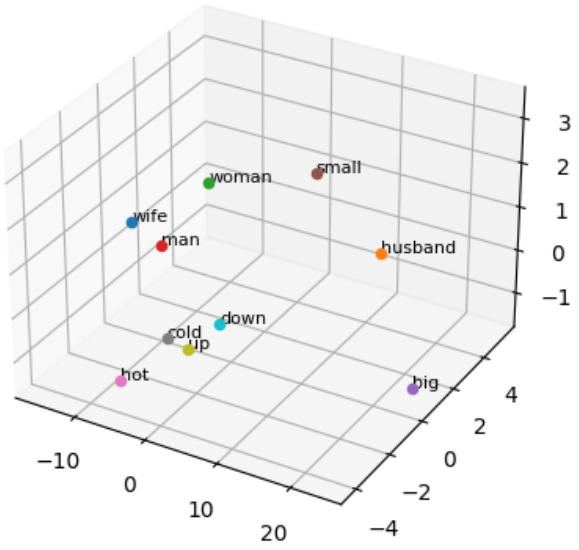

FIGURE 2.5 – L'analyse en composantes principales permet de réduire les dimensions d'un modèle en faisant un résumé de ses principaux composants. Dans l'image ci-dessus les mots *hot, cold, up, down, man, woman, husband, wife, big, small* ont été projetés dans le modèle de langue GPT-2 réduit en ses trois principaux composants. Les mots sémantiquement analogues devraient se retrouver plus près l'un de l'autre, étant donné que leurs représentations vectorielles devraient être similaires.

2.2.5.2 Word2vec

L'algorithme Word2vec permet de plonger les mots dans un espace vectoriel (Mikolov *et al.*, 2013a). Dans les domaines de la linguistique computationnelle, un n-gramme (aussi appelé *skipgram*) est une séquence contiguë de n éléments d'un échantillon donné de texte. Les éléments peuvent être des phonèmes, des syllabes, des lettres ou des mots (Chen et Goodman, 1999).

Skipgram. Word2vec basé sur des *skipgrams* à pour objectif de prévoir le contexte d'un mot compte tenu du mot lui-même. Le contexte d'un mot peut être représenté par un ensemble de paires (mot cible, mot contextuel), où un mot contextuel apparaît dans le contexte du mot cible. Par exemple, avec la phrase *le ciel est bleu*, si *ciel* est le mot cible, alors son contexte devient {(ciel, le), (ciel, est), (ciel, bleu)}.

On désigne c comme étant un mot contextuel et w comme étant le mot dans son contexte, où c et w sont présents dans l'ensemble du vocabulaire V. Étant donné un corpus de texte, l'objectif est de fixer les paramètres Θ afin que $p(c \mid w; \Theta)$ maximise la probabilité du corpus. Ici, D est l'ensemble de toutes les paires de mots et de contextes extrait du texte, C est l'ensemble de tous les contextes disponibles.

On aimerait définir les paramètres de manière à maximiser la probabilité qu'un mot contextuel soit associé à un mot donné dans un corpus. Conformément à la théorie de l'information, on a donc :

$$\Theta^* = \arg max_\Theta \prod p(c \mid w; \Theta). \tag{2.16}$$

Une approche pour calculer la probabilité conditionnelle $p(c \mid w; \Theta)$ est d'utiliser la fonction *softmax*, avec v_c et v_w qui sont respectivement des représentations vectorielles pour c et w. Notons que l'on différencie les mots de leurs contextes, de sorte que, par exemple, le vecteur associé au mot *pomme* est différent du vecteur associé au contexte *pomme*.

$$p(c \mid w; \Theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}. \tag{2.17}$$

En remplaçant le produit par la somme des logarithmes, on a donc :

$$\Theta^* = \arg max_\Theta \sum_{(w,c) \in D} \log p(c \mid w) = \sum_{(w,c) \in D} \left(\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w} \right). \tag{2.18}$$

On suppose que la maximisation de l'objectif se traduira par de bons plongements où des mots similaires seront associés à des vecteurs similaires. Cependant, la probabilité $p(c \mid w; \Theta)$ est très coûteuse à calculer en raison de la somme sur tous les contextes c'.

Échantillonnage négatif. Mikolov et al. présentent l'approche d'échantillonnage négatif comme un moyen plus efficace d'obtenir un plongement de mots (Mikolov *et al.*, 2013b). Il présente une façon de rendre le calcul plus accessible en remplaçant la fonction *softmax* avec un échantillonnage

négatif. Même si l'échantillonnage négatif est basé sur le modèle des *skipgrams*, cela optimise en fait un objectif différent. C'est-à-dire qu'on a besoin d'un mécanisme qui distingue les différentes représentations vectorielles selon leur signification, tout en interdisant certaines combinaisons (w, c).

Ainsi, on note $p(D = 1 \mid w, c)$ la probabilité que (w, c) provienne des données du corpus. Alors, une façon de faire est de présenter au modèle des paires (w, c) pour lesquelles $p(D = 1 \mid w, c; \Theta)$ est faible, c'est-à-dire des paires qui ne sont pas considérées dans les données d'entraînement. Ceci est réalisé en générant un ensemble noté D' contenant des paires (w, c) aléatoires, en supposant qu'elles sont incorrectes. Le terme *échantillonnage négatif* provient du fait qu'on tire au hasard des échantillons négatifs.

L'objectif d'optimisation devient désormais :

$$\begin{aligned}
\Theta^* &= \arg max_\Theta \prod_{(w,c)\in D} p(D = 1 \mid w; c; \Theta) \prod_{(w,c)\in D'} p(D = 0 \mid w; c; \Theta) \\
&= \arg max_\Theta \prod_{(w,c)\in D} p(D = 1 \mid w; c; \Theta) \prod_{(w,c)\in D'} (1 - p(D = 1 \mid w; c; \Theta)) \\
&= \arg max_\Theta \log \left(\prod_{(w,c)\in D} p(D = 1 \mid w; c; \Theta) \prod_{(w,c)\in D'} (1 - p(D = 1 \mid w; c; \Theta)) \right) \\
&= \arg max_\Theta \sum_{(w,c)\in D} \log p(D = 1 \mid w; c; \Theta) + \sum_{(w,c)\in D'} \log(1 - p(D = 1 \mid w; c; \Theta)) \\
&= \arg max_\Theta \sum_{(w,c)\in D} \log \frac{1}{1 + e^{-v_c \cdot v_w}} + \sum_{(w,c)\in D'} \log(1 - \frac{1}{1 + e^{-v_c \cdot v_w}}) \\
&= \arg max_\Theta \sum_{(w,c)\in D} \log \frac{1}{1 + e^{-v_c \cdot v_w}} + \sum_{(w,c)\in D'} \log \frac{1}{1 + e^{v_c \cdot v_w}},
\end{aligned} \tag{2.19}$$

avec

$$p(D = 1 \mid w; c; \Theta) = \frac{1}{1 + e^{-v_c \cdot v_w}} \tag{2.20}$$

Au lieu d'utiliser le corpus entier, il est possible de présenter l'objectif pour un exemple $(w, c) \in D$ et k exemples $(w, c_{d'}) \in D'$, en construisant D' en fonction de k. Plus précisément, D' est k fois plus grand que D, puisque pour chaque $(w, c) \in D$ nous construisons k échantillons $(w, c_1), \ldots, (w, c_k)$. La figure 2.6 illustre comment les *skipgrams* de l'ensemble D sont échantillonnés à l'aide d'une fenêtre de taille 2.

Taille de fenêtre	Texte	Skipgrams
2	[Le **ciel** est bleu] et la mer est calme.	(ciel, le), (ciel, est), (ciel, bleu)
	[Le ciel **est** bleu et] la mer est calme.	(est, le), (est, ciel), (est, bleu), (est, et)
	Le [ciel est **bleu** et la] mer est calme.	(bleu, ciel), (bleu, est), (bleu, et), (bleu, la)

FIGURE 2.6 – Adapté de (Team, 2019). La taille de la fenêtre k détermine l'étendue du contexte de chaque côté du mot cible. Les *skipgrams* hors du contexte sont considérés dans l'ensemble D' et sont étiquetés à faux. L'objectif d'échantillonnage négatif est de faire une approximation de la valeur *softmax* sur tous les contextes et consiste, à minimiser une fonction de perte selon un problème de classification, à savoir si le *skipgram* fait partie des échantillons négatifs ou non en prenant en compte le contexte établi par la fenêtre k. Prenons par exemple le premier échantillon S_1 = [(ciel, le), (ciel, et), (ciel, la)] et ses étiquettes [1 0 0] grâce auxquelles on ajuste ensuite les poids du plongement par rétropropagation en minimisant l'entropie croisée des étiquettes par rapport au vecteur $[\sigma(v_w \cdot v_c)$, pour tout $(w, c) \in S_1]$, où $\sigma =$ la fonction sigmoïde.

On fait l'hypothèse que les mots dans des contextes similaires ont des significations similaires. L'objectif ci-dessus essaie d'augmenter la quantité $v_w \cdot v_c$ pour les bonnes paires mot-contexte, et de la diminuer pour les mauvaises. Intuitivement, cela peut signifier que les mots qui partagent de nombreux contextes seront similaires les uns par rapport aux autres et que les contextes partageant de nombreux mots seront similaires de la même manières (Goldberg et Levy, 2014).

CHAPITRE 3
DICTIONNAIRE INVERSÉ NEURONAL

On s'intéresse maintenant à la construction d'un modèle capable d'associer des phrases de longueur arbitraire et représentées par des vecteurs à valeurs continues de longueur fixe afin d'établir le pont entre la sémantique lexicale et la sémantique des phrases. Les plongements de mot permettent de bâtir ce genre de modèle (Hill *et al.*, 2016; Zhang *et al.*, 2020).

De manière générale, durant la phase d'apprentissage d'un réseau neuronal pour ce genre de tâche, on retrouve principalement deux stratégies. Les mots d'une définition sont plongés dans un espace vectoriel à l'entrée d'un modèle neuronal dont la sortie : **(i)** devient l'entrée d'une couche *softmax* suite à laquelle on minimise la perte d'entropie croisée par rapport au mot cible ; **(ii)** est comparée à la représentation vectorielle du mot cible où l'on tente de diminuer la distance cosinus. Par la suite, les poids du réseau sont mis à jour par rétropropagation (Hill *et al.*, 2016; Zhang *et al.*, 2020; Kartsaklis *et al.*, 2020).

Soit y la sortie du modèle M pour la phrase P :

$$y = M(P) \tag{3.1}$$

Pour la tâche du dictionnaire inversé et afin que la sortie du modèle soit interprété en termes de probabilité, y doit être un vecteur ayant la taille du vocabulaire. Or, ce n'est pas toujours le cas pour un modèle donné. Pour remédier à cela, on peut utiliser une projection linéaire (Radford *et al.*, 2018; Devlin *et al.*, 2019). Généralement, une projection est une opération qui déplace des points dans un sous-espace. Plus précisément, une projection sur un espace vectoriel E est un opérateur linéaire $L : E \to E$ tel que $L^2 = L$. Lorsque E a un produit scalaire et est dans un espace de Hilbert [1], le concept d'orthogonalité s'applique (Strang *et al.*, 1993). En mathématiques, l'orthogonalité peut être utilisée pour désigner la séparation des caractéristiques spécifiques d'un système (Meyer, 2000).

Dans l'équation suivante, on utilise une matrice W qui projette les points du vecteur de sortie du

1. Un espace de Hilbert est un espace vectoriel muni d'un produit scalaire qui définit une fonction de distance pour laquelle l'espace est un espace métrique complet, par exemple l'espace Euclidien (Young, 1988).

modèle M dans un sous-espace de dimension de la taille du vocabulaire :

$$\begin{aligned} y &= WM(P), \\ |y| &= |V|. \end{aligned} \tag{3.2}$$

Alors, une fonction *softmax* interprète y en termes de probabilités, c'est-à-dire la probabilité qu'un symbole w soit associé à la phrase d'entrée.

$$p(w) = \text{softmax}(y_i) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|} \exp(y_j)}. \tag{3.3}$$

Afin, de maximiser la probabilité, on minimise l'entropie croisée. Formellement, le principe de minimisation de l'entropie croisée est défini par comme suit. Soit une distribution *a priori* $q(x)$. On doit déterminer la distribution $p(x)$ *la plus proche* (au sens de l'entropie croisée) qui satisfait les contraintes données (Moher et Gulliver, 1998). Conformément à la théorie de l'information, on calcule l'entropie croisée à l'aide d'une somme des logarithmes des prévisions (Shannon, 1951; Kullback, 1951).

$$\begin{aligned} q(w_i) &= \begin{cases} 1, & \text{si étiquette}(w_i) = i; \\ 0, & \text{sinon;} \end{cases} \\ J_{\text{cross}} &= -\sum_{i=1}^{|V|} q(w_i) \log(p(w_i)). \end{aligned} \tag{3.4}$$

On cherche à atteindre le minimum global de J à l'aide des dérivées partielles des poids du réseau W par rapport à J. Ensuite, on met à jour W selon un facteur α, défini comme étant le taux d'apprentissage :

$$W \leftarrow W - \alpha \frac{\partial J}{\partial W}. \tag{3.5}$$

De cette manière, Hill et al. adaptent quatre variantes de modèles neuronaux au problème du dictionnaire inversé, que nous définissons dans les sections suivantes :

— ADD, modèle basé sur Word2vec et appelé ADD, car on utilise la somme de plongements de mots ;

— BoW linéaire, modèle combinant une projection linéaire et le concept du sac de mots BoW (de l'anglais, *bag of words*) ;
— RNN, modèle basé sur les réseaux neuronaux récurrents RNN (de l'anglais, *recurrent neural network*) ;
— LSTM, modèle basé sur les réseaux récurrents avec mémoire LSTM (de l'anglais, *long short term memory*).

3.1 ADD

Hill et al. expérimentent à l'aide d'un modèle nommé ADD dans lequel on ne fait qu'additionner les plongements préentraînés Word2vec des mots d'une définition d'entrée (Hill *et al.*, 2016). Aucune phase d'apprentissage n'est nécessaire puisque les plongements de mots utilisés sont préentraînés. Certains auteurs omettent les plongements des mots fonctionnels dans la sommation, ayant observé que cela peut améliorer la précision (Morinaga et Yamaguchi, 2018).

Cela se traduit par :

$$\hat{v}_w = \sum_{w' \in P} \text{word2vec}(w'). \tag{3.6}$$

3.2 BoW linéaire

Hill et al. implémentent aussi une architecture appelée BoW linéaire afin d'encoder les définitions (Hill *et al.*, 2016). Ils entraînent le modèle M dans le but d'associer la définition d'entrée P pour le mot w à un emplacement proche du plongement v_w, au moyen d'une seule matrice de poids W. Deux fonctions de coût ont été testées soit la distance cosinus et la fonction de rang (en anglais, *rank loss*).

Le modèle a pour objectif d'associer les plongements des mots de la définition d'entrée avec le plongement du mot cible u. Alors, l'estimation du vecteur cible v est calculé à l'aide d'une somme de projections linéaires des plongements des mots associés à la définition P :

$$v = \sum_{w' \in P} W v_{w'}. \tag{3.7}$$

Conformément à la distance cosinus expliquée à la section 2.2.4, on a

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}, \tag{3.8}$$

$$J_{\text{distance}}(u, v) = 1 - \cos(u, v). \tag{3.9}$$

La fonction de rang a pour but de discriminer un symbole cible par rapport à des symboles choisis aléatoirement. On veut que $\cos(u, v)$ soit plus grand que $\cos(v, r)$ par une marge m, pour tout autre symbole dans le vocabulaire, où r est un plongement de mot choisi au hasard autre que celui du mot cible (Huang *et al.*, 2012). Une marge m fixée à 0.1 s'est avérée efficace pour des tâches de récupération de mots (Bordes *et al.*, 2015) :

$$J_{\text{rank}}(u, v, r) = \max(0, m - \cos(u, v) - \cos(v, r)). \tag{3.10}$$

3.3 Réseau neuronal récurrent

Les réseaux neuronaux récurrents (RNN) ont été proposés pour pallier une lacune des réseaux de neurones *feed-forward*, qui ont tendance à mal gérer les données séquentielles et ne mémorisent pas les entrées précédentes (Biswal, 2022). Pour ce faire, les RNN utilisent des cellules possédant une mémoire interne qui se met à jour selon l'état d'activation de la cellule précédente.

La figure 3.1 illustre le processus de récurrence dans un réseau RNN.

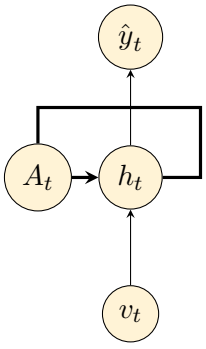

FIGURE 3.1 – Adapté de (Helmini, 2019). Les connexions des RNN sont récurrentes, suivant une approche qui s'alimente elle-même. Chacune des récursions calcule un état caché h. L'état caché au temps t est calculé à l'aide d'une fonction d'activation A_t, de l'état caché au temps $t - 1$ et de l'entrée actuelle v_t. Ce processus permet de conserver des informations sur ce que le modèle a observé auparavant.

Dans le cas de la tâche du dictionnaire inversé, pendant la formation, l'entrée du RNN est une définition de dictionnaire. L'état final du réseau est un résumé des informations contenues dans la phrase d'entrée. L'objectif du modèle est d'associer l'état final au plongement Word2vec du mot cible.

Au temps $t = 1$, le RNN prend comme entrée le plongement du premier mot de la phrase P et on lui applique une projection non linéaire paramétrée une matrice de poids W, et un biais b. Cela produit l'état d'activation interne A_1. Hill et al. utilisent la fonction d'activation $\phi(x) = \tanh(x)$, bien que ϕ puisse être n'importe quelle autre fonction non linéaire différentiable. L'état au temps $t+1$ est mis à jour en additionnant la projection de l'état au temps t, paramétrée avec U, une autre matrice de poids.

Cela se traduit par les équations :

$$\begin{aligned} A_1 &= \phi(Wv_1 + b), \\ A_t &= \phi(UA_{t-1} + Wv_t + b). \end{aligned} \tag{3.11}$$

3.4 *Long short-time memory*

Dans le domaine du traitement de la langue, l'utilisation telle quelle des RNN peut être limitée, puisque les gradients obtenus par la méthode de descente de gradient sont susceptibles de décliner en *disparaissant* (leur norme s'approche de 0) ou en *explosant* (leur norme tendent vers $+\infty$) à mesure que le nombre de pas de temps (longueur de la phrase) augmente (Bengio *et al.*, 1994). Par conséquent, lors du traitement de phrases plus longues, l'activation interne finale conserve généralement des informations utiles sur les mots les plus récemment lus, mais tend à négliger des informations importantes qui se situent au début de la phrase d'entrée.

Prenons un exemple où les dérivées partielles des mises à jour $\frac{\partial L}{\partial A}$ reposent sur le calcul du gradient de la fonction sigmoïde. Le problème de cette fonction se produit lorsque la sortie est proche de 0 ou 1 puisque les gradients sont proches de 0. Étant donné que les gradients deviennent pratiquement nuls au fil du temps, les poids ne s'ajustent plus et, par conséquent, le réseau ne converge pas (Dhruv et Naskar, 2020).

Les réseaux de neurones LSTM ont été conçus pour répondre à ce problème (Hochreiter et Schmidhuber, 1997). À chaque pas de temps t, on modifie la couche d'activation en calculant six états internes i^w, g^i, g^f, g^o, h et m.

— i^w représente les informations de base du mot courant qui sont transmises à la cellule LSTM au temps t ;
— h_{t-1} représente les informations de base du mot courant qui sont transmises à la cellule LSTM au temps $t-1$;
— m_t représente la mémoire interne d'une cellule LSTM au temps t ;
— g^f détermine dans quelle mesure m est conservée ou oubliée ;
— g^i détermine dans quelle mesure le nouveau mot d'entrée est pris en compte à chaque pas de temps ;
— g^o détermine combien cette mémoire est prise en compte lors du calcul de l'état de sortie à t.

D'abord, on initialise les vecteurs h et m avec des valeurs aléatoires contenues entre 0 et 1. Ensuite, on projette linéairement i avec quatre matrices de poids W_w, W_i, W_f, W_o et h_{t-1} avec quatre matrices de poids U_w, U_i, U_f et U_o. Les entrées d'une cellule sont calculées en prenant les matrices $W_w + U_w$, $W_i + U_i$, $W_f + U_f$ et $W_o + U_o$.

Les états g^i, g^f et g^o sont ensuite calculés à l'aide de fonctions sigmoïdes de leur projection d'entrée associée. Ces vecteurs prennent des valeurs dans $[0, 1]$ et sont souvent appelés *activations des portes* (en anglais, *gating activation*). Le prochain état de sortie h_t est calculé en faisant le produit de la fonction tanh de la mémoire interne m_t par la sigmoïde de l'état d'entrée i^w.

Cela se traduit par les équations :

$$
\begin{aligned}
& s \in \{i, o, f\}, \\
& i_t^w = W_w v_t + U_w h_{t-1} + b_w, \\
& i_t^s = W_s v_t + U_s h_{t-1} + b_s, \\
& g_t^s = \sigma(i_t^s) = \frac{1}{1 + \exp(-(i_t^s))}, \\
& m_t = \phi(i_t^w) \times g_t^i + m_{t-1} \times g_t^f, \\
& h_t = g_t^o \times \phi(m_t).
\end{aligned}
\tag{3.12}
$$

L'état final du réseau LSTM, $h_{|P|}$, est un résumé des informations contenues dans la phrase P. À partir de cet état, on cible l'espace de plongement associé au mot recherché à l'aide d'une projection linéaire, puis on met à jour les poids du réseau par rétropropagation.

La figure 3.2 illustre le flux d'information dans les différentes portes d'activation d'une cellule d'un réseau LSTM.

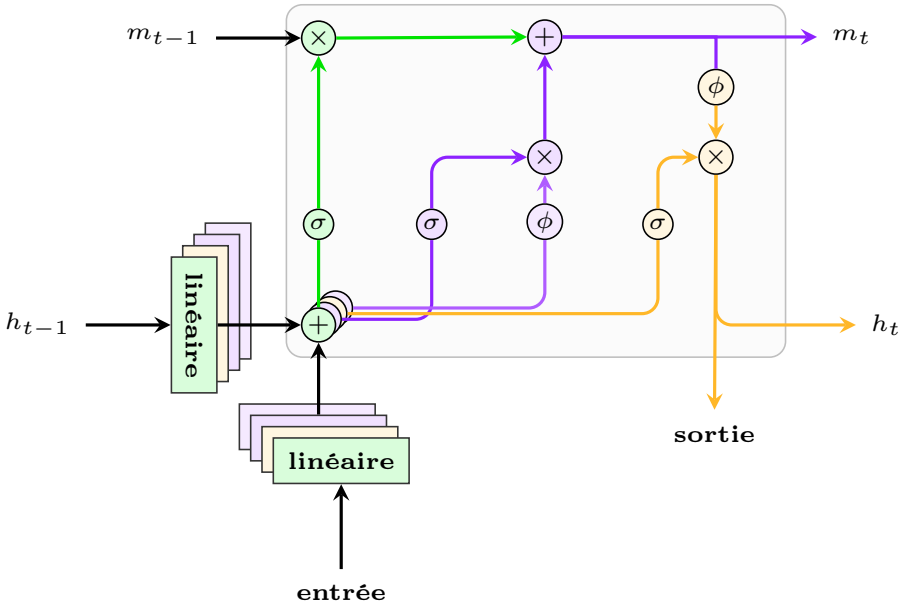

FIGURE 3.2 – Adapté de (Kienzler, 2017). Au temps t, une cellule LSTM contient un état caché h_t et une mémoire m_t. Au temps $t = 1$, les vecteurs h_1 et m_1 sont initialisés aléatoirement. Les zones verte, pourpre et jaune représentent respectivement les portes d'oubli g_t^f, d'entrée g_t^i et de sortie g_t^o. L'intuition derrière cette représentation est qu'en multipliant le résultat de la sortie de la porte d'oubli avec le résultat de la mémoire interne au temps précédant, une partie de l'information sera effacée de la mémoire du réseau m_t à mesure que les pas de temps progressent. Autrement dit, au début, les valeurs du vecteur résultant de la porte d'oubli se rapprochent de 1, et la quasi-entièreté de l'information de l'entrée est maintenue. En progressant dans le temps, le vecteur est mis à jour en effaçant ou en retenant de l'information.

3.5 Convolution sur texte

Morinaga et Yamaguchi expérimentent à l'aide d'un réseau de neurones convolutif de classification de texte [2] (Kim, 2014) permettant de faire des prévisions sur les 45 catégories possibles que propose WordNet pour un mot cible (Morinaga et Yamaguchi, 2018). Ce modèle est utilisé comme filtre des sorties des modèles de Hill. On définit C comme étant une matrice de vecteurs de caractéristiques (en anglais, *features map*) et $p(c \mid P)$ la probabilité avec laquelle une entrée donnée P est associée à une catégorie. On définit aussi i noyaux (en anglais, *kernels*), qu'on note W_i et qui ont une fenêtre (en anglais, *receptive field*) de taille $k_i \times |E|$. Les vecteurs de caractéristiques sont calculés à l'aide de convolutions, c'est-à-dire qu'on superpose un noyau à la matrice de plongement de mots E, des rangés j à $j + k_i$ inclusivement, où j est initialisé à zéro puis incrémenté $|P| - k_i + 1$ fois. Ensuite,

2. `https://github.com/harvardnlp/sent-conv-torch`

on calcule la somme des multiplications des valeurs du noyau avec ses valeurs associées dans E. Les vecteurs de caractéristiques passent ensuite par des opérations de *max pooling* (la valeur maximale pour un vecteur de caractéristique), de concaténation et sont finalement linéairement connectés à une couche de prévision *softmax*.

Cela se traduit par les équations :

$$\begin{aligned} C_{i,j} &= \phi(W_i E_{j:j+k-1} + b_i), \\ \text{pool}_i &= \max(C_i), \\ y &= \text{linéaire}(pool), \\ p(c|P) &= \frac{\exp(y_c)}{\sum \exp(y)}. \end{aligned} \tag{3.13}$$

La figure 3.3 illustre le mécanisme de convolution pour quatre noyaux sur une matrice de plongements de jetons associée à la phrase « le ciel est bleu et la mer est bien calme », dans le but de catégoriser celle-ci parmi deux classes.

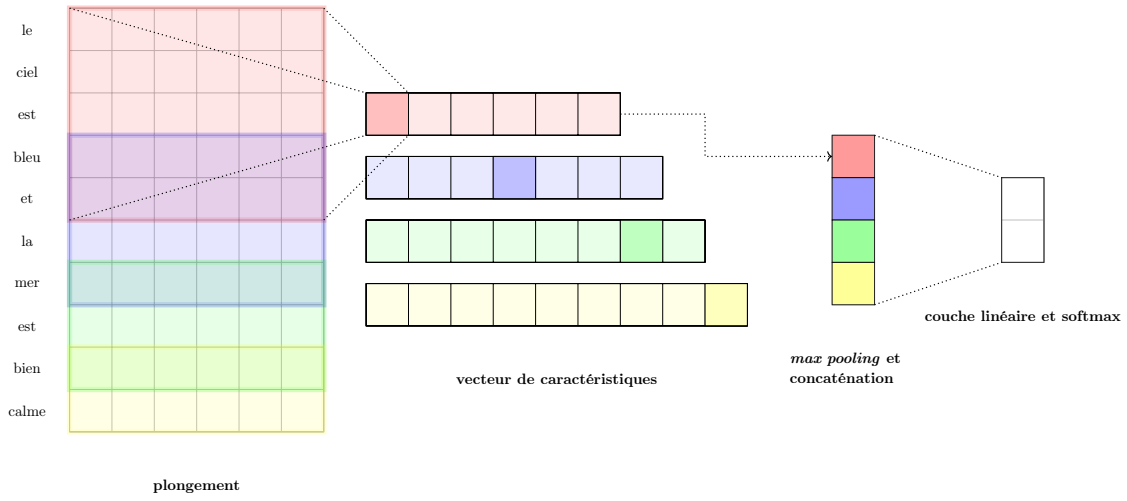

FIGURE 3.3 – Les noyaux rouge, bleu, vert et jaune ont respectivement une fenêtre de taille 5, 4, 3 et 2 ainsi qu'un indice j associé égal à 0, 3, 6 et 8. Il se trouve que l'indice j désigne l'indice de convolution dans un vecteur de caractéristiques pour un noyau donné. L'avantage d'utiliser plusieurs noyaux réside dans le fait qu'en faisant varier la valeur de k sur plusieurs noyaux, on est en mesure de capter des relations sémantiques dans une fenêtre de k mots d'une phrase.

Un modèle BoW linéaire ou LSTM filtre alors ses prévisions à l'aide du réseau de neurones convolutif de classification de texte formé. S'il y a une catégorie ayant une probabilité supérieure à 0.95, on choisit les mots candidats appartenant à cette catégorie (Morinaga et Yamaguchi, 2018).

3.6 Réseau de neurones *Cascade Feed-Forward*

Dans une étude, Morinaga et Yamaguchi ont utilisé un réseau de neurones *feed-forward* (FNN) composé de 5 couches complètement connectées (Morinaga et Yamaguchi, 2018). Le réseau de neurones *Cascade Feed-Forward* (CFNN) est un FNN auquel on ajoute des connexions supplémentaires (en anglais, *bypass structures*) entre les couches non adjacentes. Le CFNN est similaire au réseau DenseNet (Huang *et al.*, 2017), cependant il n'a pas de couches de *pooling* ni de couche *softmax*, car sa sortie est utilisée comme une approximation du vecteur cible.

La figure 3.4 illustre comment on raccorde les connexions supplémentaires d'un réseau CFNN ainsi que son flux de données.

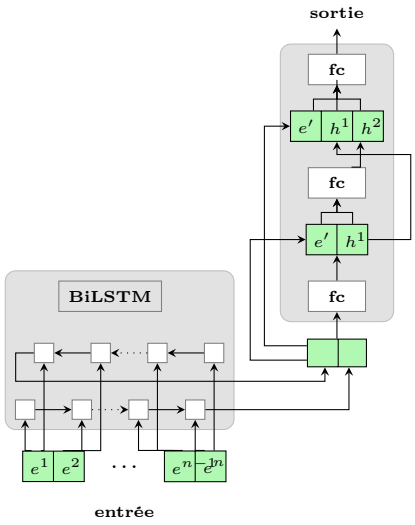

FIGURE 3.4 – Le réseau BiLSTM prend les plongements des mots en entrée et concatène son contexte de gauche et de droite. Le résultat devient l'entrée d'un réseau s'apparentant au DenseNet dans lequel on introduit des connexions supplémentaires, et où les couches précédentes sont concaténées aux couches suivantes. Ces connexions diminuent le risque que les gradients *disparaissent* ou *explosent*, tout en ayant un réseau profond.

Pour bien former ce réseau, les auteurs notent deux conditions essentielles afin de bien former le

réseau. Le LSTM et le réseau multicouche entièrement connecté doivent être entraînés simultanément et le LSTM ne doit pas avoir trop de capacité.

3.7 Dictionnaire inversé multicanal

La connaissance du POS (voir section 1.1.2), du morphème, de la catégorie de mot ou du sémème facilite la recherche du mot cible. Un modèle de Zhang utilise des prédicteurs identifiant ces caractéristiques (Zhang *et al.*, 2020). De cette manière, les mots cibles ayant un poids défavorable dans le plongement peuvent être tout de même sélectionnés. Le modèle filtre aussi les mots ayant un plongement similaire, mais de caractéristiques contradictoires. Aussi, les différents mots d'une phrase ont chacun une importance différente par rapport au sens de la phrase. Un mécanisme d'attention est un mécanisme qui attribue un score à chacun des mots d'une phrase afin de pondérer leur importance (Bahdanau *et al.*, 2015). Ce mécanisme est expliqué plus en détail à la section 4.1.2.

Ainsi, le dictionnaire inversé multicanal est une architecture LSTM bidirectionnelle avec mécanisme d'attention sur laquelle on greffe quatre prédicteurs de caractéristiques. La figure 3.5 illustre le flux d'information dans le modèle multicanal.

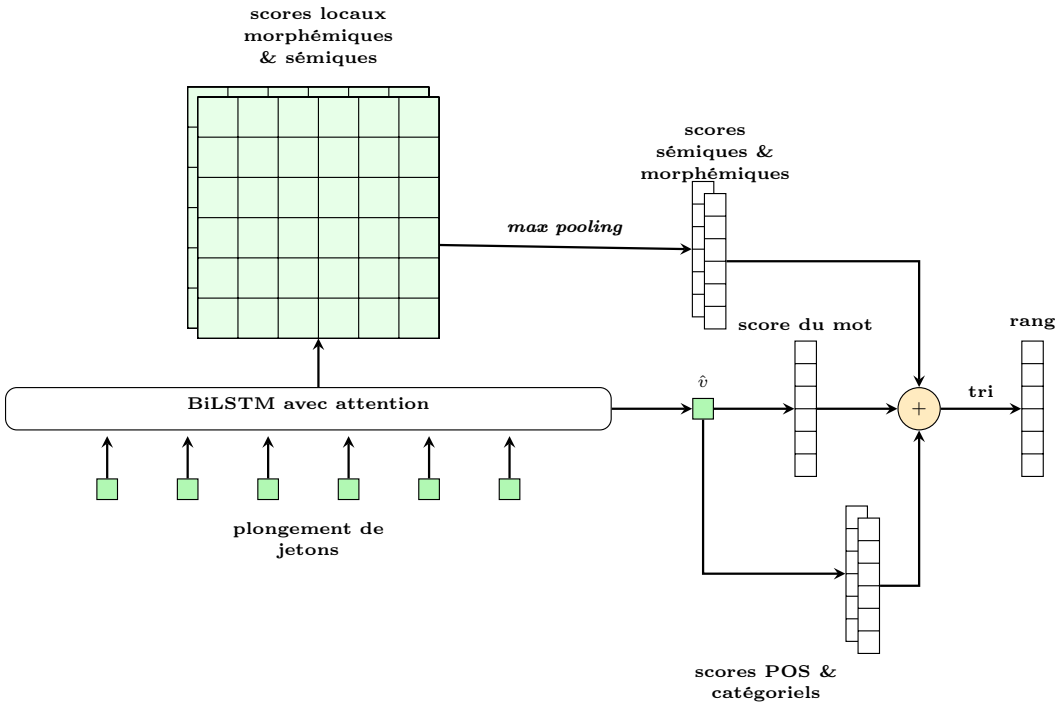

FIGURE 3.5 – Adapté de (Zhang *et al.*, 2020). Un réseau BiLSTM est un réseau LSTM qui concatène les contextes gauche et droit des plongements d'entrée. Le réseau calcule des vecteurs associés aux attributs sémiques, morphémiques et catégoriels des mots cibles. Pour les vecteurs des scores sémiques et morphémiques, on effectue des projections linéaires sur ses états cachés, puis un *max pooling* afin de cibler les morphèmes et sémèmes du mot recherché. Pour obtenir un estimé vectoriel du mot cible, on additionne les états cachés pondérés par leur mécanisme d'attention. Ensuite, on effectue des projections linéaires sur celui-ci afin de cibler la catégorie et le POS du mot recherché. Enfin, l'addition des vecteurs résultants devient l'entrée d'une couche *softmax* qui calcule les probabilités qu'un mot du vocabulaire y soit associé.

Score d'un mot. Le modèle prend comme entrée le plongement préentraîné associé à la description et génère deux séquences d'états cachés directionnels. Leur concaténation se solde par des états cachés non directionnels. La représentation d'une définition est donnée par une somme pondérée en fonction des états cachés non directionnels et à l'aide du mécanisme d'attention.

Cela se traduit par les équations :

$$
\begin{aligned}
&\{\vec{h_1}, \ldots, \vec{h_{|P|}}\}, \{\overleftarrow{h_1}, \ldots, \overleftarrow{h_{|P|}}\} = \text{BiLSTM}(P), \\
&h_i = \text{concatenate}(\vec{h_i}, \overleftarrow{h_i}), \\
&h_t = \text{concatenate}(\vec{h_{|P|}}, \overleftarrow{h_i}), \\
&\alpha_i = h_t \cdot h_i, \\
&\hat{v} = \sum_{i}^{|P|} \alpha_i h_i,
\end{aligned}
\tag{3.14}
$$

où

— α_i : score du mécanisme d'attention pour le i-ème mot d'un phrase P

.

Prédicteur morphémique. Certains mots ont un morphème qui est sémantiquement lié à des termes de leur définition. Par conséquent, leur prévision capture des informations structurantes du mot cible. Chaque état caché non directionnel alimente un perceptron simple couche (PSC) qui projette linéairement ceux-ci afin d'obtenir des scores locaux. Pour chaque mot, on choisit le maximum parmi leurs scores locaux pour déterminer leur score morphémique global.

Prédicteur sémique. Les sémèmes peuvent décrire avec précision le sens du mot. Par exemple, pour le mot *autoroute*, on a deux possibles sémèmes avec *route* et *auto* qui correspond sémantiquement à une définition $P = \langle \text{chemin, voiture} \rangle$. Afin de capturer cela, on projette linéairement chaque état caché à l'aide d'un PSC calculant les scores de sémèmes locaux. Pareils au score morphémique, les scores finaux sémiques sont calculés en fonction des scores locaux maximums.

Prédicteur catégoriel. Les informations sur les catégories de mots permettent de filtrer les résultats des mots sémantiquement liés sans être analogues, par exemple *route* et *voiture*. À partir d'un arbre d'analyse, une catégorie et un POS sont étiquetés à un mot cible. Un PSC calcule les scores de la catégorie et du POS.

Prédicteur de synonyme. Malekzadeh et al. propose un modèle en langue perse basé sur cette architecture et auquel ils ajoutent un prédicteur de synonyme (Malekzadeh *et al.*, 2021).

3.8 *Multi-sense* LSTM

Kartaklis et al. expérimentent sur la composition de phrases dans un espace multidimensionnel obtenu à partir d'un graphe de connaissances. Un graphe KB (de l'anglais *knowledge base*) est un graphe qui comprend des nœuds de caractéristiques textuelles qui agissent comme des ponts sémantiques avec des nœuds d'entité (mot) (Kartsaklis *et al.*, 2020).

Pour chaque mot, on collecte les descriptions textuelles trouvées dans de grands dictionnaires intégrant des ressources variées, telles que BabelNet, WordNet, Wikipédia et FrameNet (Miller, 1995; Baker *et al.*, 1998; Navigli et Ponzetto, 2012). Les descriptions textuelles sont traitées comme des documents courts où chaque mot se voit attribuer une valeur TF-IDF spécifique, formant l'ensemble des caractéristiques textuelles. Le graphe KB est étendu de la manière suivante : Soit T_c l'ensemble des caractéristiques textuelles pour une entité w. Pour chaque t dans T_c, on ajoute un arc (w, t) de poids tf-idf w_t, où tf-idf w_t est la valeur TF-IDF de t à l'égard de w. Soit un nœud n sélectionné au hasard, C_n l'ensemble de tous les nœuds d'entité dans son voisinage immédiat et T_n l'ensemble de toutes les caractéristiques textuelles de n. Dans un chemin, le nœud x suivant est tiré d'une distribution catégorielle :

$$p(x) = \begin{cases} (1-\lambda)\frac{1}{|C|} & \text{si } x \in C_n, \\ \lambda \frac{\text{tf-idf}_n(t_i)}{\sum_{j=1}^{|T|} \text{tf-idf}_n(t_j)} & \text{si } x \in T_n. \end{cases} \tag{3.15}$$

Kartsaklis et al. utilisent des chemins aléatoires à partir d'un graphe KB (Kartsaklis *et al.*, 2020), qui est ensuite utilisé comme entrée dans un modèle de type *skipgram* (Mikolov *et al.*, 2013a).

La figure 3.6 illustre comment la valeur de λ affecte l'échantillonnage des nœuds pour un chemin issu de la base de connaissance.

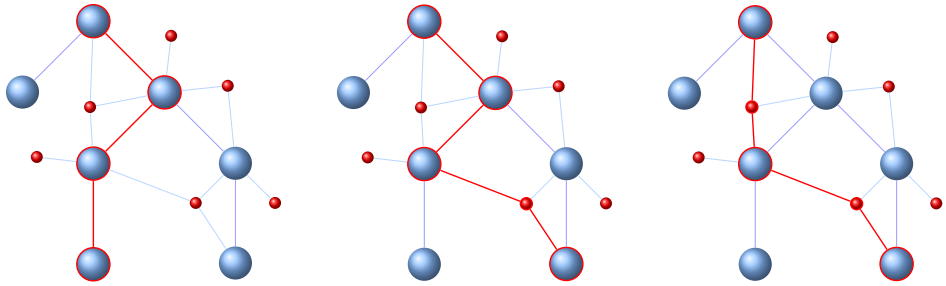

FIGURE 3.6 – Adapté de (Kartsaklis *et al.*, 2020). Pour un graphe KB, assigner $\lambda = 0$ entraîne un processus d'échantillonnage qui ignore les caractéristiques textuelles (nœuds rouges) et produit un chemin composé uniquement de nœuds d'entité (nœuds bleus). En revanche, assigner $\lambda = 1$ crée des chemins suivant un modèle alterné, où chaque nœud d'entité est suivi d'un nœud textuel, qui à son tour est suivi d'un nœud d'entité.

Pour un chemin aléatoire de nœuds $n_1, n_2 \dots n_T$, pour un nœud cible n_t, l'objectif est de prédire tous les autres nœuds contenus dans une fenêtre contextuelle k.

$$J = \frac{1}{T} \sum_{t=1}^{T} \sum_{-k \leq j \leq k, j \neq 0} \log p(n_{t+j} \mid n_t). \tag{3.16}$$

En conséquence, deux vecteurs seront proches si leurs nœuds correspondants se trouvent à proximité topologique dans le graphe. Cependant, bien qu'une telle topologie permette des comparaisons significatives entre des points dans l'espace du plongement, elle n'est pas directement compatible avec la tâche d'association de texte sur des mots. Les relations formées dans ce modèle reflètent principalement des hiérarchies spécifiques au domaine et des relations ontologiques qui ne sont pas nécessairement relatives aux représentations textuelles faisant référence aux mots.

Architecture. Le modèle comprend un plongement fixe de mots, une couche de désambiguïsation du sens des mots et deux couches LSTM et une base de connaissance sous forme de graphe. La sortie de l'attention est une somme pondérée des vecteurs de sens compte tenu de leurs probabilités, qui est utilisée comme entrée au réseau LSTM à 2 couches. Ensuite, avec l'erreur quadratique moyenne MSE (pour, en anglais *mean square error*), on compare les vecteurs de sortie du LSTM aux vecteurs cibles sortant de la base de connaissance.

Afin de pallier le problème de polysémie, pour chaque mot w_i dans un exemple d'apprentissage, un

vecteur de contexte c_i est calculé comme la moyenne des vecteurs du plongement fixe des autres mots de la phrase. La probabilité de chaque vecteur de sens s_{ij}, compte tenu de ce contexte, est alors calculée via un mécanisme d'attention suivi d'une couche *softmax*. Chaque vecteur de sens est ensuite mis à jour en additionnant son vecteur contexte pondéré par sa similarité avec le sens spécifique.

Cela se traduit par les équations :

$$\begin{aligned} s'_{ij} &= \phi(W s_{ij} + U c_i), \\ p(s_{ij}|c_i) &= \frac{\exp(w'_j \cdot s'_{ij})}{\sum_l^k \exp(w'_l \cdot s'_{il})}, \\ s_{ij}^{t+1} &= s_{ij}^t + (s_{ij}^t \cdot c_i^t) c_i^t. \end{aligned} \tag{3.17}$$

L'objectif est de minimiser l'erreur quadratique moyenne entre les vecteurs prédits et les vecteurs cibles, où N est le nombre d'exemples d'apprentissage, x le texte d'entrée, y le vecteur du mot cible et f le réseau de neurones. L'erreur quadratique moyenne est alors donnée par :

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} ||y_i - f(x_i)||^2. \tag{3.18}$$

Les auteurs traitent WordNet comme un graphe dont les arcs sont définis par les différentes relations entre les *synsets*. Ce graphe comprend 96 734 nœuds textuels extraits des descriptions de *synset*. On fixe $k = 150$, sur des marches aléatoires de longueur 20 et avec une taille de fenêtre de 5. Pour l'évaluation des données vues, on forme le réseau LSTM sur la totalité des *synsets* WordNet et leurs descriptions. Pour l'évaluation des données non vues, on supprime du graphe toutes les caractéristiques textuelles apparaissant dans la partie de test et on crée un nouvel ensemble de vecteurs *synset*. En outre, toute instance de test est supprimée de l'ensemble d'apprentissage. L'évaluation se fait en comparant le vecteur prédit avec les vecteurs de tous les *synsets* WordNet et en créant une liste classée comme précédemment, par similarité cosinus.

La figure 3.7 illustre le flux d'information dans un modèle *multi-sense* et plus particulièrement son mécanisme d'attention.

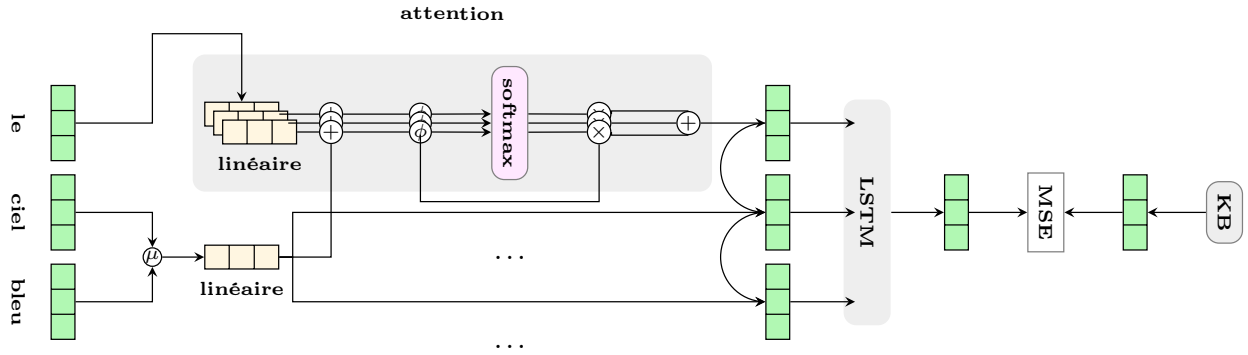

FIGURE 3.7 – Adapté de (Kartsaklis *et al.*, 2020). Afin de capturer les attributs sémantiques d'un mot, on utilise un mécanisme d'attention qui ajoute, à n projections du mot courant, une projection linéaire de la moyenne μ des plongements du contexte c du mot courant. On fait la somme des projections pondérées à l'aide d'une fonction *softmax*. Les représentations sémantiques sont ensuite mises à jour en tenant compte de leurs contexte. Ici, le mot courant est *le*, $n = 3$, et $\mu = \frac{v_{\text{ciel}} + v_{\text{bleu}}}{n}$. L'intuition derrière ce mécanisme est que l'on obtient une représentation sémantique du mot qui capte partiellement les relations contextuelles qu'apporte chacun des autres mots de l'entrée.

3.9 Onelook

Onelook est un modèle qui résout le problème du dictionnaire inversé. Onelook est exploité comme application commerciale en ligne et son architecture est propriétaire. Certaines des fonctionnalités de OneLook sont activées par une analyse statistique des mots d'une vaste collection de livres écrits au cours du siècle dernier. Il indexe 1061 dictionnaires ainsi que des ressources telles que Wikipédia et WordNet (Hill *et al.*, 2016). Notons que l'algorithme est capable de faire un tri des associations de mots qui reflètent des stéréotypes racistes ou nuisibles et que les entrées peuvent contenir des mots génériques (en anglais, *wildcard*).

Onelook utilise aussi l'API (en anglais, *Application Programming Interface*) Datamuse, qui utilise à son tour plusieurs ressources linguistiques .

3.9.1 Datamuse

Datamuse offre un large éventail de fonctionnalités. Par exemple, pour un mot donné et certaines contraintes programmables, l'API peut retourner des synonymes, des mots relatifs, ou encore des mots qui riment (Datamuse, 2023).

Données phonétiques. Le dictionnaire de prononciation CMU est utilisé comme source pour les différentes relations phonétiques de la langue anglaise telles que les rimes et les homophones.

Données basées sur le corpus. L'ensemble de données Google Books Ngrams est utilisé pour analyser les relations lexicales entre les mots et afin de construire un modèle de langue qui note les mots candidats en fonction du contexte.

Connaissance sémantique. WordNet ainsi que des dizaines de dictionnaires en ligne sont utilisés afin d'analyser les relations sémantiques entre les mots.

Dans la littérature scientifique, Onelook est surtout utilisé à titre comparatif de performance entre les modèles. En effet, on le retrouve comme étalon de référence dans de nombreuses études (Hill *et al.*, 2016; Zhang *et al.*, 2020; Yan *et al.*, 2020; Kartsaklis *et al.*, 2020).

CHAPITRE 4
TRANSFORMEURS

Les transformeurs ont été introduits dans le contexte de la traduction automatique dans le but d'éviter la récursivité et afin de faciliter le calcul parallèle. Ils permettent de réduire le temps d'entraînement et également de mitiger les mauvaises performances dues aux dépendances de longues portées (Vaswani *et al.*, 2017). Les principales caractéristiques des transformeurs incluent : **(i)** un modèle non séquentiel qui traite les phrases dans leur ensemble, **(ii)** un mécanisme d'auto-attention qui calcule les scores de similarité entre les mots d'une phrase, **(iii)** un encodage positionnel qui encode les informations liées aux positions des jetons dans une phrase.

Les transformeurs ne s'appuient pas sur les états cachés passés afin de capturer les dépendances entre les mots. Ils traitent plutôt une phrase dans son ensemble à l'aide d'un mécanisme d'auto-attention et d'un encodage positionnel qui fournissent des informations sur la relation entre les mots.

4.1 Architecture originale

L'architecture originale d'un transformeur comprend une pile d'encodeurs et une pile de décodeurs du même nombre. Pour le premier encodeur, l'entrée est le plongement de mots auxquels on ajoute l'encodage positionnel, c'est-à-dire un signal qui indique l'ordre des mots dans la séquence d'entrée. Pour les couches subséquentes, l'entrée est la sortie de l'encodeur qui se trouve au-dessous (Vaswani *et al.*, 2017).

Durant l'encodage, les donnés passent par une couche d'auto-attention, c'est-à-dire une couche qui aide l'encodeur à prendre en compte d'autres mots dans la phrase d'entrée lorsqu'il encode un mot spécifique. Les sorties de la couche d'auto-attention sont transmises à une couche linéaire puis vers l'encodeur suivant. La sortie de la pile d'encodeurs est transmise aux décodeurs et ceux-ci font remonter leurs résultats de décodage, similairement à l'encodeur, jusqu'à ce qu'un symbole spécial soit atteint par la dernière couche *softmax* connectée à la sortie de la pile de décodeurs, indiquant que le transformeur a terminé le traitement (Vaswani *et al.*, 2017).

La figure 4.1 illustre le flux d'information dans l'architecture originale d'un transformeur et plus

particulièrement la manière dont l'information est transmise de la pile d'encodeurs vers la pile de décodeurs.

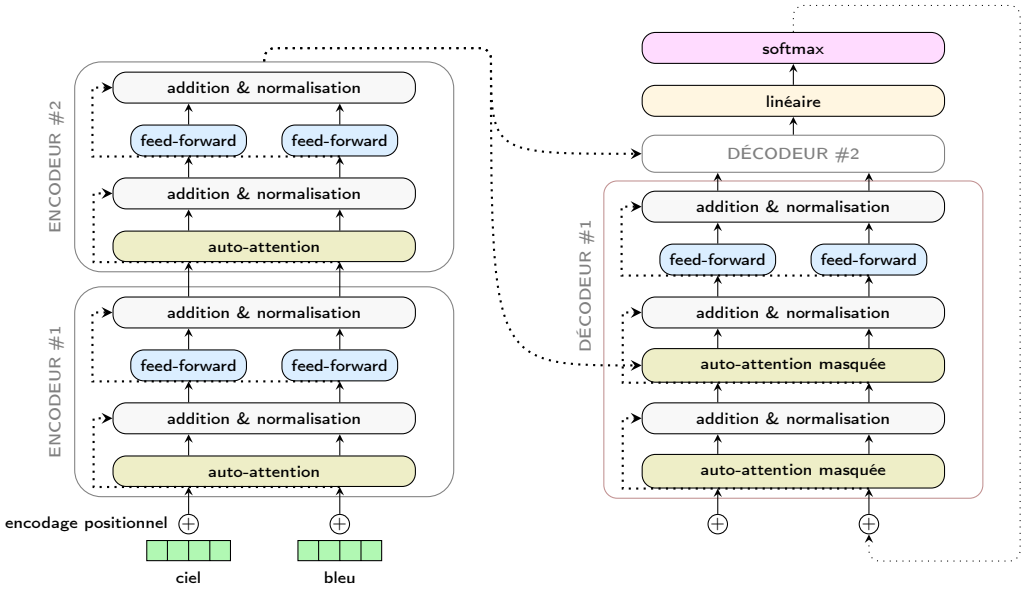

FIGURE 4.1 – Adapté de (Jalamar, 2020b). La couche *softmax* permet de générer des jetons qui s'ajoutent au fil du temps, jusqu'à ce qu'on génère un jeton spécial `EOS` qui indique que le modèle a terminé le traitement.

4.1.1 Encodage positionnel

Le modèle tel que décrit jusqu'à présent doit trouver un moyen de tenir compte de l'ordre des mots dans la séquence d'entrée. Contrairement aux modèles séquentiels comme les RNN et LSTM, les transformeurs ne reposent pas sur un mécanisme de capture des positions relatives des mots dans une phrase. Ceci est important, car la distance entre les mots fournit des informations contextuelles (Jalamar, 2020b).

Pour ce faire, on utilise un encodage positionnel (en anglais, *positional embedding*). L'encodage positionnel ne fait pas partie de l'architecture du modèle, mais plutôt du prétraitement et est généré de manière à avoir la même taille que celui du plongement. L'intuition derrière l'encodage positionnel vient du fait que cela fournit des distances significatives dans les projections linéaires effectuées dans les différentes couches du réseau (Bahdanau *et al.*, 2015).

Ainsi, on utilise une fonction PE qui prend en compte la position du mot par rapport à l'indice pos

d'un mot de la séquence, de le l'indice i du vecteur de plongement et du nombre de dimensions des plongements de mots d. En calculant le sinus et le cosinus de l'angle, on retourne une valeur pour un mot à la position *pos* selon l'indice i du vecteur du plongement. Pour un mot de la séquence, la valeur de *pos* reste constante à mesure que l'indice du plongement i augmente. En passant au mot suivant, on incrémente *pos* et cela décale légèrement le motif vers la droite. Les fonctions sinus et cosinus sont périodiques alors, peu importe la longueur de la séquence d'entrée, les encodages positionnels auront toujours des valeurs dans l'intervalle $[-1, 1]$ (Vaswani *et al.*, 2017).

Cela se traduit par les équations :

$$\begin{aligned} \text{angle} &= \frac{\text{pos}}{n^{2i/d}}, \\ \text{PE}_{(pos,2i)} &= \sin(\text{angle}), \\ \text{PE}_{(pos,2i+1)} &= \cos(\text{angle}), \end{aligned} \tag{4.1}$$

où

- d : nombre de dimensions des plongements de mots,
- *pos* : indice d'un mot de la séquence d'entrée,
- i : indices des colonnes de l'encodage positionnel associés aux valeurs de retour des fonctions sinus et cosinus avec $0 \leq i < d/2$,
- n : scalaire fixé à 10 000 par les auteurs du papier original (Vaswani *et al.*, 2017).

Exemple.

Prenons un exemple avec la séquence d'entrée $P = \langle le, ciel, est, bleu \rangle$ où $n = 100$ et $d = 4$.

		Encodage Positionel			
entrée	**pos**	$i = 0$	$i = 0$	$i = 1$	$i = 1$
le	0	sin(0)	cos(0)	sin(0)	cos(0)
ciel	1	sin(1/1)	cos(1/1)	sin(1/10)	cos(1/10)
est	2	sin(2/1)	cos(2/1)	sin(2/10)	cos(2/10)
bleu	3	sin(3/1)	cos(3/1)	sin(3/10)	cos(3/10)

FIGURE 4.2 – Adapté de (Saeed, 2023). L'encodage positionel est une matrice de dimensions $(|P|, d)$. Pour tout i dans l'intervale $[0, \ldots, d/2[$, on retourne les valeurs de la fonction $\sin(\frac{pos}{n^{2i/d}})$ aux indices $(pos, 2i)$ et les valeurs de la fonction $\cos(\frac{pos}{n^{2i/d}})$ aux indices $(pos, 2i + 1)$.

Après avoir calculé le plongement positionnel, celui-ci est ajouté au plongement d'entrée.

4.1.2 Mécanisme d'attention

Disons qu'une phrase d'entrée est : « Le chat traverse le pont, car il a faim ». Dans cet exemple, *il* est une anaphore pour le mot *chat* et non pour le mot *pont*. L'objectif du mécanisme d'attention est donc d'associer le mot *il* au mot *chat*. Au fur et à mesure que le modèle traite chacun des mots, le mécanisme d'attention permet au modèle de porter attention à différentes positions dans la séquence d'entrée afin de trouver des indices qui conduisent à un meilleur encodage pour ce mot. De façon analogue aux RNN, en particulier la façon dont ils maintiennent un état caché permettant au modèle d'incorporer la représentation des mots précédents au mot traité couramment, le mécanisme d'attention est une méthode utilisée par le transformeur afin d'intégrer une compréhension des autres mots pertinents par rapport au mot traité couramment (Jalamar, 2020b).

La figure 4.3 illustre l'intuition derrière l'importance que donne le mécanisme d'attention à certains mots pour la phrase « Le chat traverse le pont, car il a faim ».

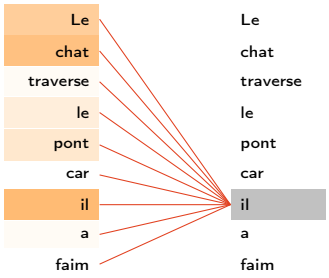

FIGURE 4.3 – Adapté de (Jalamar, 2020b). Les mots de couleurs foncées auront plus d'importance que les mots plus pâles. C'est-à-dire qu'à ce stade, pour le mot *il*, l'algorithme accordera des poids plus élevés pour lui-même, le mot *Le* et le mot *chat*.

La première étape du calcul de l'auto-attention consiste à créer trois vecteurs à partir des vecteurs d'entrée de l'encodeur. Pour chaque mot, on crée un vecteur **Requête** q, un vecteur **Clé** k et un vecteur **Valeur** v en projetant linéairement le plongement de mot à l'aide de trois matrices W_q, W_k et W_v.

1. **Requête** : Les vecteurs de requête sont des représentations du mot courant.
2. **Clé** : Les vecteurs clés représentent des *étiquettes* pour tous les mots de la séquence d'entrée. On les utilise comme éléments de comparaison lorsqu'on recherche les mots pertinents par rapport à une requête.
3. **Valeur** : Les vecteurs de valeur sont des représentations *réelles* des mots. Une fois qu'on a évalué la pertinence de chaque mot, on ajoute les vecteurs de valeurs afin de représenter les mots courants.

La figure 4.4 illustre comment on calcule les valeurs des vecteurs **K**, **Q** et **V**.

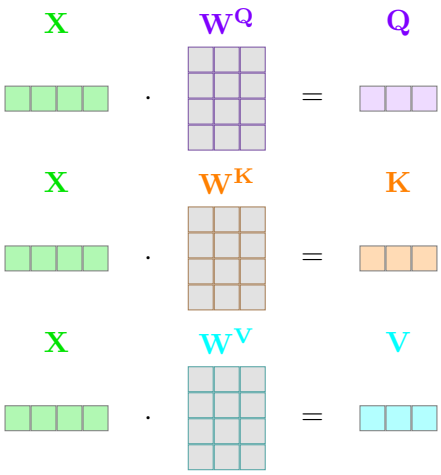

FIGURE 4.4 – Adapté de (Jalamar, 2020b). Pour obtenir les vecteurs **K**, **Q** et **V**, on projette une entrée X avec trois matrices de poids.

La deuxième étape consiste à calculer un score pour chaque mot de la phrase d'entrée par rapport au mot couramment traité afin de déterminer l'importance à accorder aux autres mots de la phrase d'entrée pour un mot à une autre position. Les scores de chacun des mots sont calculés avec le produit scalaire du vecteur q du mot courant et les vecteurs k des autres mots.

La troisième étape consiste à normaliser le résultat en divisant les scores par la racine carrée de la dimension du modèle. Vaswani et al. affirment que pour de grandes valeurs de d , les produits scalaires augmentent en amplitude, poussant la fonction *softmax* dans des régions où elle a des gradients extrêmement petits. Pour contrer cet effet, on met à l'échelle les produits scalaires (Vaswani *et al.*, 2017).

Quatrièmement, on effectue une opération *softmax* sur le résultat. Le mot courant à une certaine position aura le score *softmax* le plus élevé par rapport à lui-même. Cependant, il est utile de s'occuper d'un autre mot pertinent par rapport au mot courant (Jalamar, 2020b).

La cinquième étape consiste à multiplier chaque vecteur v par le score *softmax*. L'intuition derrière cette étape est de garder intactes les valeurs des mots sur lesquels on veut se concentrer, et de diminuer l'importance des mots moins pertinents en les multipliant par de petits nombres (Jalamar, 2020b).

La sixième étape consiste à faire la somme des vecteurs pondérés. Cela produit la sortie de la couche d'auto-attention pour le mot courant.

La figure 4.5 illustre les différentes étapes du mécanisme d'attention pour le premier mot d'une suite de deux mots.

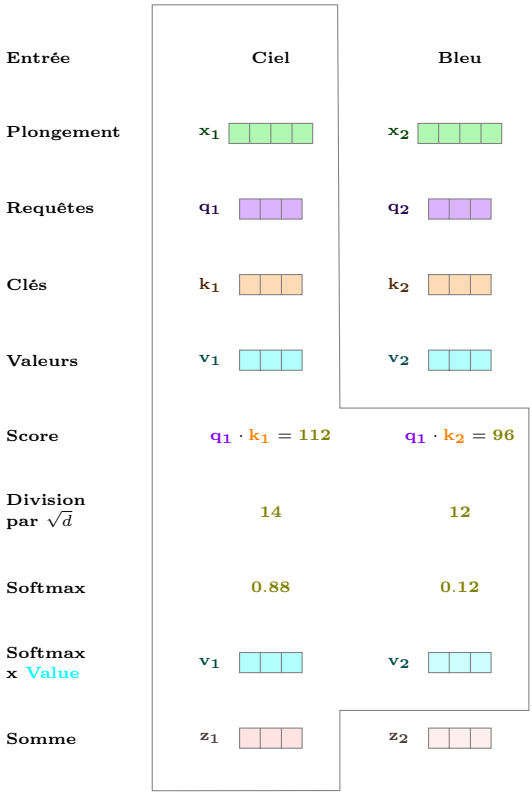

FIGURE 4.5 – Adapté de (Jalamar, 2020b). Après avoir calculé les valeurs de $\mathbf{Q}$, $\mathbf{K}$ et $\mathbf{V}$, on effectue une série d'opérations arithmétiques afin d'ajuster le mécanisme d'attention. L'intuition est que l'on obtient un score en comparant chaque mot q à chaque mot k avec lequel on pondère la valeur réelle v d'un mot.

Il est possible de simplifier toutes ces étapes avec un calcul matriciel :

$$Z = \text{softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^\mathsf{T}}{\sqrt{d}}\right) \cdot \mathbf{V}. \tag{4.2}$$

4.1.3 Multitêtes

La sortie de la couche d'attention contient une partie de tous les autres encodages, mais elle peut être dominée par le mot courant. Une attention *multitêtes* permet d'étendre la capacité du modèle à se focaliser sur différentes positions en utilisant plusieurs ensembles de matrices $\mathbf{Q}/\mathbf{K}/\mathbf{V}$. L'avantage est que cela se traduit par de multiples sous-espaces de représentation. Or, la couche suivante du modèle n'attend pas plusieurs matrices, mais plutôt une seule matrice représentant un vecteur pour chaque mot. Pour ce faire, on concatène les matrices puis on les multiplie par une matrice de poids supplémentaire W^z.

La figure 4.6 illustre le fonctionnement du mécanisme multitêtes et l'étape de redimensionnement de la matrice de sortie.

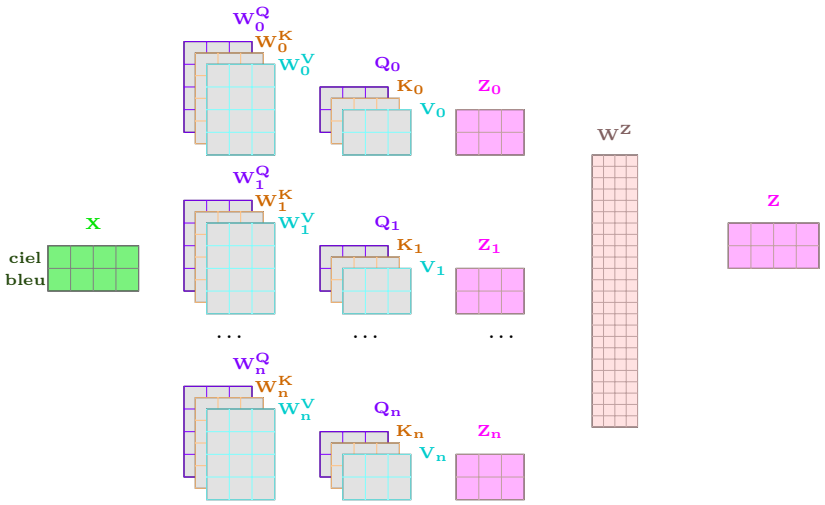

FIGURE 4.6 – Adapté de (Jalamar, 2020b). Pour le premier encodeur, les mots *ciel* et *bleu* représentent les entrées auxquels on associe un plongement X. On aura ensuite n multitêtes contenant chacune trois matrices W afin de produire le mécanisme d'attention Z en calculant les matrices Q, K et V pour chacune des têtes. On doit reconvertir les vecteurs Z calculés par les différentes têtes en une matrice de même taille que l'entrée. Pour ce faire, on projette linéairement la concaténation des Z avec une autre matrice de poids W^z. Pour les encodeurs suivants, l'entrée est le résultat d'une normalisation de l'addition de Z et de X auquel on applique une transformation non linéaire (en anglais, *feed-forward*).

4.1.4 Mécanisme d'attention masqué

Le mécanisme d'attention du décodeur fonctionne d'une manière légèrement différente de celle de l'encodeur. Les couches d'attention créent leurs matrices **Q** à partir de la couche précédente et prennent les matrices **K** et **V** à partir de la sortie de la pile d'encodeurs. De plus, la couche du mécanisme d'attention du décodeur n'est pas autorisée à regarder les positions antérieures dans la séquence de sortie. Cela se fait en masquant les positions futures avant l'étape du *softmax*. Ce masquage peut prendre la forme d'une matrice appelée masque d'attention dans laquelle on insère des valeurs de $-\infty$ dans le triangle supérieur.

La figure 4.7 illustre comment le mécanisme d'attention du décodeur masque les entrées.

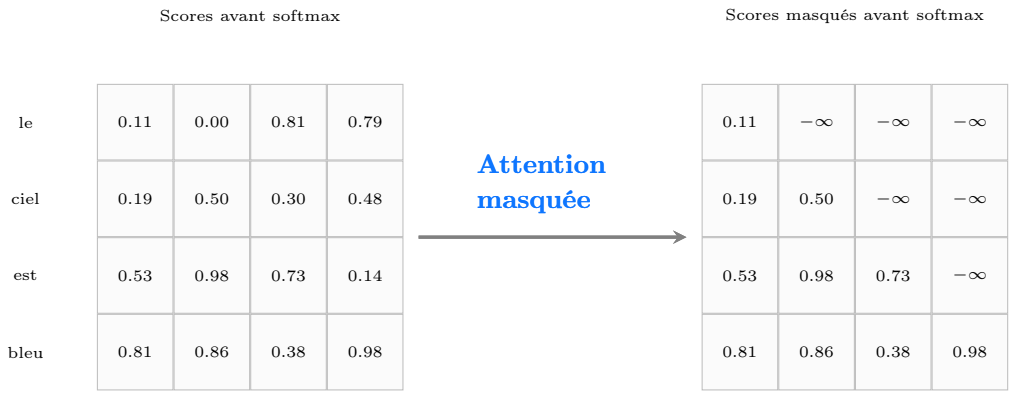

FIGURE 4.7 – Adapté de (Jalamar, 2020a). Dans le décodeur, l'attention masquée consiste à cacher les mots succédant au mot courant et seulement prendre en compte les mots qui lui précèdent. Prenons par exemple la phrase d'entrée $P = \langle$le,ciel,est,bleu$\rangle$, et des scores aléatoires. Après l'opération du masque, le mot *le* ne portera plus attention aux mots *ciel*, *est* et *bleu*. Quant à lui, le mot *bleu* portera attention à tous les mots qui lui précèdent.

4.1.5 Prévision du prochain jeton

La pile du décodeur retourne un vecteur de dimension d à chaque pas de temps. Une couche linéaire (voir section 3.2) projette les vecteurs retournés par la pile de décodeurs dans d'autres vecteurs de dimensions de la taille du vocabulaire. Supposons que le modèle connaisse 10 000 jetons uniques tirés des données d'entraînement. La couche linéaire retournerait des vecteurs de 10 000 dimensions à chaque pas de temps et où chaque dimension a un score associé à un jeton unique. La couche *softmax* (voir section 3.3) transforme ensuite ces scores en probabilités. Pour un pas de temps

donné, la dimension ayant la probabilité la plus élevée est choisie et le jeton associé est retourné en sortie pour alimenter l'entrée au prochain pas de temps. Le modèle génère des jetons jusqu'à ce que le jeton spécial `<EOS>` (pour en anglais, *end of sentence*) soit prédit.

4.1.6 Phase d'apprentissage

Pendant la formation, puisqu'on génère un jeton de sortie à la fois, on utilise la méthode d'entraînement par instruction forcée (en anglais, *teacher-forcing*), où on alimente les jetons d'étiquette (plutôt que ceux prédits dans la couche de *softmax*), ce qui rend l'apprentissage plus stable. Cela rend également la formation plus rapide, car on peut préparer les masques d'attention, permettant un traitement parallèle (Irissappane *et al.*, 2020). Le modèle ajuste ensuite ses paramètres par rétropropagation en minimisant l'entropie croisée (voir section 3.4).

4.1.7 Transfert d'apprentissage

Dans le cadre du TAL, l'apprentissage des transformeurs relève globalement de la catégorie de l'apprentissage semi-supervisé. La préformation non supervisée est un cas particulier d'apprentissage semi-supervisé où le but est de trouver un bon point d'initialisation (Radford *et al.*, 2018). Des approches récentes ont étudié l'apprentissage automatique de la sémantique au niveau des mots à partir de données non étiquetées. Les plongements de mots entraînés à l'aide d'un corpus non étiqueté ont été utilisés afin de résoudre, par la suite, diverses tâches cibles du TAL (Kiros *et al.*, 2015; Le et Mikolov, 2014).

La préformation des réseaux de neurones peut aussi agir comme un schéma de régularisation, permettant une meilleure généralisation de leurs prévisions (Ramachandran *et al.*, 2017). Dans des travaux récents, la méthode a été utilisée afin d'aider à former des réseaux de neurones profonds sur diverses tâches telles que la désambiguïsation des entités (He *et al.*, 2013) et la traduction automatique (Ramachandran *et al.*, 2017). L'axe de travail consiste à préformer un réseau de neurones à l'aide d'un objectif de modélisation du langage puis à l'affiner sur une tâche cible avec supervision (Hill *et al.*, 2016; Radford *et al.*, 2018).

4.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin *et al.*, 2019) est un modèle bâti à partir d'une pile d'encodeurs provenant de l'implémentation originale du transformeur (Vaswani *et al.*, 2017). Le modèle est conçu pour préentraîner des représentations bidirectionnelles à partir de texte non étiqueté en conditionnant conjointement les contextes gauche et droit dans toutes les couches (Devlin *et al.*, 2019). L'architecture atténue la contrainte d'unidirectionnalité en utilisant un modèle de langue masqué (MLM, de l'anglais *masked language model*), où on masque au hasard certains des jetons de l'entrée avec l'objectif de prédire l'identifiant du symbole masqué en se basant uniquement sur son contexte. En fusionnant les contextes gauche et droit grâce au mécanisme d'attention, on forme un transformeur multicouche bidirectionnel profond. En conséquence, BERT peut être affiné (en anglais, *fine-tuned*) en ajoutant une couche de sortie afin de créer des modèles destinés à résoudre un éventail de tâches.

4.2.1 Entrées

Pour générer chaque séquence d'entrée d'apprentissage, on échantillonne deux parties du texte du corpus, qu'on appelle des *phrases*, même si elles sont généralement beaucoup plus longues que des phrases simples (mais peuvent également être plus courtes).

BERT prend 3 types d'entrée.

Plongement des jetons. Au lieu d'utiliser directement les mots d'une phrase, on utilise ce qu'on appelle la segmentation de sous-mots pour décomposer les mots complexes en mots simples, puis les convertir en jetons. Cela fait en sorte qu'on utilise les contextes de sous-mots de mots complexes. Pour ce faire, BERT utilise WordPiece (voir section 1.1.1). Les phrases sont échantillonnées de telle sorte que la longueur combinée est d'au plus 512 jetons.

Plongement de segments. Les plongements de segments sont utilisés afin d'aider BERT à distinguer les différentes phrases dans une entrée. Les valeurs de ce vecteur sont identiques pour tous les mots d'une même phrase. La valeur change si la phrase est différente. Supposons qu'on a traité les deux phrases $P_1 = \langle$Le, ciel, est, bleu$\rangle$ et $P_2 = \langle$et, la, mer, est, calme$\rangle$. On segmentera les phrases de façon à placer un jeton `[CLS]` au début de P_1, des jetons `[SEP]` au début et à la fin de P_2. Le

plongement de segments associé sera pourvu de valeurs égales à 0 pour tous les éléments associés à P_1 et de valeurs égales à 1 pour tous les éléments correspondant à P_2.

Jetons de masque. Pour rendre les mots segmentés compatibles avec la taille d'entrée, on les aligne par remplissage (en anglais, *padding*). Ainsi, les jetons de masque aident BERT à différencier les mots modifiés aux mots à traiter. Puisque le modèle accepte une entrée de 512 jetons, on génèrera un masque de taille 512 dans lequel les indices correspondant aux mots à traiter auront des valeurs de 1 et les indices correspondant au *padding* auront des valeurs de 0.

Implémentation. Pour résumé, on segmente les phrases P_1 et P_2 en plaçant un jeton `[CLS]` au début de P_1, des jetons `[SEP]` au début et à la fin de P_2, des jetons `[MASK]` suivant une procédure de masquage décrite dans une section plus bas et des jetons `[PAD]` pour les k jetons manquant qui complètent les 512 jetons d'entrée qu'attend le modèle. En langage Python, cela s'exprime par :

```
p1 = ['[CLS]', 'le', 'ciel', 'est', '[MASK]']
p2 = ['[SEP]', 'et', 'la', 'mer', 'est', 'calme', '[SEP]']
k = 512 - len(p1) - len(p2)
padding = ['[PAD]'] * k
jetons = p1 + p2 + padding
segment = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1] + [1] * k
masque = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] + [0] * k
```

4.2.2 Sortie

Pour les tâches de classification, les sorties de tous les états cachés associées à la séquence d'entrée ne sont pas nécessaires, car BERT ne considère que la sortie correspondant au premier jeton `[CLS]`. Cela fait en sorte que lorsque le modèle est préentraîné, ce jeton encode des informations globales spécifiques à la phrase d'entrée. Il serait aussi possible d'utiliser une moyenne des derniers états cachés de la séquence d'entrée au lieu de la sortie du jeton `[CLS]`, mais dans ce cas, on doit omettre les jetons de *padding* dans le calcul de la moyenne.

4.2.3 Tâches de prévision

Prévision du masque. On choisit au hasard 15% des positions des jetons masqués à prédire. Soit $P = \langle$mon, chien, est, poilu$\rangle$, une phrase aléatoire sans étiquette. Si au cours de la procédure de masquage, le jeton *poilu* est choisi, la procédure de masquage pour ce jeton devient :

— 80% du temps : remplacer par un jeton `[MASK]`, $P = \langle$mon, chien, est, `[MASK]`$\rangle$

— 10% du temps : remplacer par un mot aléatoire, $P = \langle$mon, chien, est, pomme$\rangle$

— 10% du temps : garder le mot, $P = \langle$mon, chien, est, poilu$\rangle$.

L'avantage de cette procédure est que le modèle ne sait pas quels mots on lui demandera de prévoir ou lesquels ont été remplacés par des mots aléatoires. Il doit donc conserver une représentation contextuelle distributionnelle de chaque jeton d'entrée. Les vecteurs finaux sont introduits dans une couche *softmax* sur la taille du vocabulaire avec lequel le modèle tente de minimiser l'entropie croisée de sa prévision face à la valeur cible.

Prévision de la phrase suivante. Pour la tâche de prévision de la phrase suivante, on a comme entrée deux phrases P_1 et P_2. Ensuite, on procède comme suit : **(i)** 50% du temps P_2 est une phrase tirée du corpus qui suit P_1, et **(ii)** 50% du temps P_2 est une phrase aléatoire. Le tableau 4.1 illustre comment on étiquette les données pour la tâche de prévision de la phrase suivante.

Entrée	**Étiquette**
`[CLS] un homme va [MASK] magasin` `[SEP] il prend un litre [MASK] lait [SEP]`	Vrai
`[CLS] un homme va [MASK] magasin` `[SEP] le [MASK] est bleu [SEP]`	Faux

TABLE 4.1 – Adapté de (Devlin *et al.*, 2019). La tâche de prévision de la phrase suivante est une tâche de classification binaire où on cherche à estimer la probabilité qu'une phrase P_2 suive une phrase P_1. Soit $P_1 = \langle$un, homme, va, au, magasin$\rangle$, $P_2 = \langle$il, prend, un, litre, de, lait$\rangle$ et $P3 = \langle$le, ciel, est, bleu$\rangle$, où P_2 suit P_1 dans le corpus et où P_3 a été tirée aléatoirement dans le corpus. On génère des données de façon à ce que 50% du temps un échantillon composé de P_2 suivant P_1 est étiqueté à *vrai* et 50% du temps un échantillon est composé d'une phrase aléatoire suivant P_1 et est étiqueté à *faux*. Par exemple, un échantillon étiqueté à *faux* serait composé de P_3 suivant P_1. Notons qu'il y a d'abord un prétraitement des phrases afin que celles-ci soient adaptées au format d'entrée du modèle. De plus, la tâche de prévision de la phase suivante se fait au même moment que la tâche de prévision de jetons de masques.

On calcule une perte de classification binaire depuis une projection linéaire sur le dernier état caché

du jeton [CLS]. Cette perte est ajoutée à celle qu'on obtient en prédisant les masques.

Afin de faire des prévisions, BERT a besoin que certains jetons soient masqués, c'est-à-dire remplacés par le jeton spécial [MASK]. Dans les modèles de langue masqués, comme BERT, chaque prévision de jeton masqué est conditionnée par le reste des jetons de la phrase. La sortie est générée de manière non autorégressive c'est-à-dire que les prévisions des jetons de masque sont calculées au même moment. Le décodeur n'est donc pas nécessaire (Noe, 2019).

La figure 4.8 illustre comment BERT joint les contextes gauches et droits dans la pile d'encodeurs.

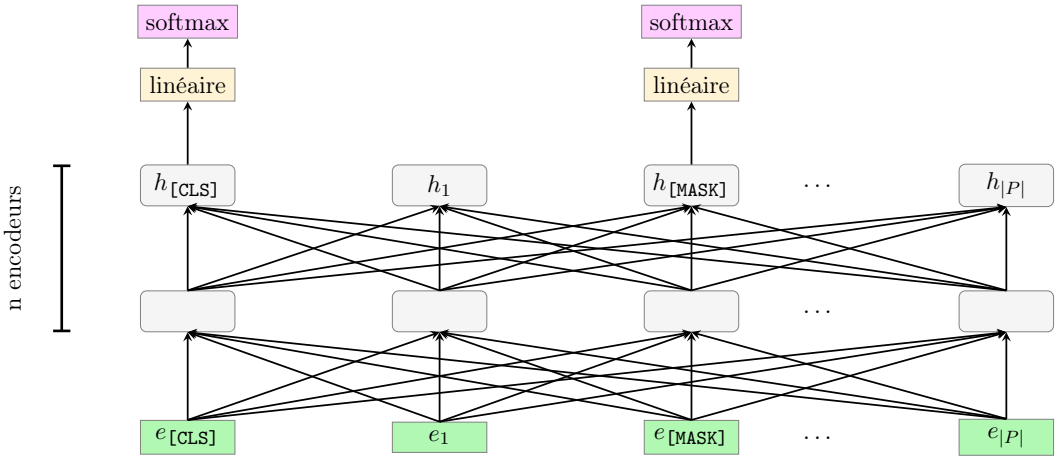

FIGURE 4.8 – Adapté de (Devlin *et al.*, 2019). BERT est un transformeur bidirectionel basé sur une pile d'encodeurs, où les contextes gauche et droit de la séquence d'entrée sont joints. La séquence d'entrée est plus précisément constituée des plongements des jetons d'une phrase, auxquels on additionne l'encodage positionnel et le plongement de segment. Chaque encodeur retourne des états cachés représentant chacun des jetons de la séquence. À partir des états cachés retournés par le dernier encodeur, le modèle minimise une fonction de perte pour tous les jetons de type [MASK]. Aussi, le dernier état caché du jeton [CLS] est employé afin de faire des prévisions sur des tâches de classifications et qui, à terme, formera une représentation générale de la phrase d'entrée.

4.2.4 Optimiseur

L'algorithme *Adaptive Moment Estimation* (Adam) est un algorithme qui optimise la descente de gradient associée à l'équation 3.5 concernant la mise à jour des poids du réseau de neurones telle qu'expliquée au chapitre 3. L'optimiseur Adam implique une combinaison de deux méthodologies de descente de gradient, c'est-à-dire le *momentum* et la propagation RMS (en anglais, *root mean square propagation*).

Momentum. Le *momentum* consiste à considérer une estimation de la moyenne pondérée exponentiellement des gradients afin d'accélérer l'algorithme de descente de gradient.

Cela se traduit par les équations :

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1-\beta_1)\left(\frac{\partial J}{\partial \Theta_t}\right), \\ \Theta_{t+1} &= \Theta_t - \alpha m_t, \end{aligned} \tag{4.3}$$

où

- Θ_t : poids au temps t,
- α : taux d'apprentissage au temps t (généralement 0.001),
- $\frac{\partial L}{\partial \Theta_t}$: dérivée partielle de la fonction de coût par rapport aux poids au temps t,
- m_t : estimation de la somme de gradients passés au temps t (initialement, $m_t = 0$),
- β_1 : paramètre de moyenne mobile (généralement constant à 0.9).

Propagation RMS. La propagation RMS consiste à adapter le taux d'apprentissage en considérant la moyenne mobile exponentielle des gradients.

Cela se traduit par les équations :

$$\begin{aligned} v_{t+1} &= \beta_2 v_t + (1-\beta_2)\left(\frac{\partial J}{\partial \Theta_t}\right)^2, \\ \Theta_{t+1} &= \Theta_t - \frac{\alpha_t}{\sqrt{v_t}+\varepsilon}\left(\frac{\partial J}{\partial \Theta_t}\right), \end{aligned} \tag{4.4}$$

où

- Θ_t : poids au temps t,
- α : taux d'apprentissage au temps t,
- $\frac{\partial L}{\partial \Theta_t}$: dérivée partielle de la fonction de coût par rapport poids au temps t,
- v_t : estimation de la somme des carrés des gradients passés au temps t (initialement, $v_t = 0$),
- β_2 : paramètre de moyenne mobile (généralement une constante à 0.999),
- ϵ : une petite constante positive.

Correction. Étant donné que m_t et v_t ont tous deux été initialisés à 0, ils ont tendance à être

biaisés vers 0, car β_1 et $\beta_2 \approx 1$. On corrige leur biais en utilisant les équations :

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t}. \end{aligned} \tag{4.5}$$

Finalement, l'optimiseur Adam combine les deux méthodes afin d'optimiser la descente de gradient :

$$\Theta_{t+1} = \Theta_t - \hat{m}_t \left(\frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \right). \tag{4.6}$$

Régularization L2. Le surapprentissage se produit lorsque le modèle performe bien sur les données d'entraînement, mais ne généralise pas bien avec des données sur lesquelles le modèle n'a pas été entraîné. Une manière de prévenir cela consiste à minimiser la perte et la complexité du modèle, ce qu'on nomme la minimisation du risque structurel, (en anglais *structural risk minimization*) (Cortes *et al.*, 2009). L'optimisation de l'apprentissage est désormais fonction de deux termes : la fonction de coût, qui mesure la performance du modèle face aux données et un terme de régularisation, qui mesure la complexité du modèle.

$$J(\Theta) \leftarrow J(\Theta) + \text{complexité}(\Theta). \tag{4.7}$$

Il est possible de quantifier la complexité du modèle à l'aide de la formule de régularisation L2, qui définit le terme de régularisation comme la somme des carrés de tous les paramètres, en partant avec l'hypothèse que les poids de plus grande magnitude augmentent la complexité du modèle (Cortes *et al.*, 2009).

Cela se traduit par l'équation :

$$\text{termes de régularisation L2} = \sum_{i}^{|\Theta|} \Theta_i^2. \tag{4.8}$$

On définit un facteur λ désignant le taux de régularisation. Si λ est élevé, le modèle est pénalisé davantage et peut avoir tendance à sous-ajuster les paramètres. En revanche, si λ est petit, l'effet

de la régulation diminue et le modèle risque de surapprendre. Si λ est paramétré à zéro, la régularisation est complètement supprimée. On multiplie λ par $\frac{1}{2}$ puisque cela n'affecte pas l'objectif de minimisation tout en simplifiant la dérivée de la fonction.

Cela se traduit avec les équations :

$$\begin{aligned} J(\Theta) &\leftarrow J(\Theta) + \frac{\lambda}{2}\sum_{i}^{|\Theta|}\Theta_i^2, \\ \frac{\partial J}{\partial \Theta_t} &= \frac{\partial J}{\partial \Theta_t} + \lambda\Theta_t. \end{aligned} \tag{4.9}$$

La mise à jour des paramètres, telle qu'expliquée à l'équation 3.5, est alors modifiée pour prendre en compte la régularisation et est donnée avec l'équation :

$$\Theta_{t+1} = \Theta_t - \alpha\frac{\partial J}{\partial \Theta_t} - \alpha\lambda\Theta_t = (1-\alpha\lambda)\Theta_t - \alpha\frac{\partial J}{\partial \Theta_t}. \tag{4.10}$$

Décroissance des poids. D'autre part, Hanson et al. introduisent le concept de décroissance des poids (en anglais, *weight decay*) comme une méthode menant à une optimisation de la phase d'apprentissage (Hanson et Pratt, 1988). L'équation concernant la décroissance des poids comme mentionnée dans l'étude de Hanson et al. est donnée par

$$\Theta_{t+1} = (1-\lambda)\Theta_t - \alpha\frac{\partial J}{\partial \Theta_t}. \tag{4.11}$$

Dans une descente de gradient stochastique sans *momentum*, on remarque que lorsqu'on paramètre $\lambda \leftarrow \frac{\lambda}{\alpha}$, la régularisation L2 et la décroissance des poids sont réciproques. Cependant, pour Adam, ce n'est pas le cas puisque les gradients ont une influence sur les paramètres m et v, alors il n'est pas possible de paramétrer λ de façon à ce que la régularisation L2 soit réciproque à la décroissance des poids (Loshchilov et Hutter, 2018).

Adam avec décroissance découplée des poids (AdamW). L'optimiseur Adam avec décroissance découplée des poids (AdamW, de l'anglais, *Adam with decoupled weight decay*) propose d'améliorer la régularisation de Adam en découplant le taux d'apprentissage et la décroissance du poids

lors de la mise à jour basée sur le gradient, puisqu'on y démontre qu'Adam généralise mieux avec la décroissance découplée des poids qu'avec la régularisation L2 (Loshchilov et Hutter, 2018).

La mise à jour des poids avec l'optimiseur Adam, telle qu'expliquée à l'équation 4.6 est remplacée par l'équation :

$$\Theta_{t+1} = \Theta_t - \eta_t \left(\frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \Theta_t \right). \tag{4.12}$$

où η peut être fixe ou adapté selon une méthode suivant une phase de réchauffement (en anglais, *warmup*), suivie d'une décroissance à chaque pas d'apprentissage, et où λ est adapté selon la taille des donnés et le nombre de pas d'apprentissage selon l'équation :

$$\lambda \leftarrow \lambda \sqrt{\frac{b}{BT}} \tag{4.13}$$

avec

- b : taille des lots (en anglais, *batch*),
- B : nombre total d'exemples d'entraînement,
- T : nombre total d'époques (nombre de fois que la totalité des données d'entraînement est vue).

4.2.5 Paramètres

Lors de la phase de préentraînement de BERT, Devlin at al. paramètrent l'optimiseur AdamW avec $\alpha = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 0.01$, le nombre de pas de réchauffement à 10 000 suivis d'une décroissance linéaire paramétrée à 0.0004, une taille de lot à 256 et un taux de *dropout*[1] fixé à 0.01.

4.2.6 Données d'apprentissage

Lors du préentraînement du modèle, les auteurs utilisent les données situées dans le BooksCorpus, constituées d'environ 800M de mots et, dans la version anglaise de Wikipédia, qui contient environ 2 500M de mots (Zhu *et al.*, 2015). Pour ce qui est des données tirées de Wikipédia, les auteurs extraient uniquement les passages de texte et ignorent les listes, les tableaux ainsi que les en-têtes.

1. Le terme *dropout* fait référence à la *suspension* de neurones dans un réseau et est expliqué plus en détail au chapitre 5

4.2.7 Word2Vec vs BERT

BERT offre un avantage par rapport à Word2vec, puisqu'il adapte sa représentation d'un mot en prenant en compte son contexte. Prenons, par exemple, deux phrases : « Mon frère a mal à la tête » et « Mon frère est en tête de file ». Étant donné ces deux phrases, l'algorithme Word2vec produirait la même représentation du mot *tête*, alors qu'avec BERT, la représentation du mot *tête* est différente dans chacune des phrases. En plus de capturer la polysémie, le plongement que produit BERT peut aussi capturer d'autres formes d'informations qui se traduisent par des représentations de caractéristiques plus précises (Shen et Liu, 2021).

4.2.8 Modèles basés sur BERT

Depuis la sortie de BERT, les modèles de langues ont évolué et pour ce faire, certains sont basés sur BERT.

ALBERT. En raison des limitations de la mémoire des GPU et du temps nécessaire à la phase d'entraînement, augmenter davantage la capacité du modèle devient problématique. Pour résoudre ces problèmes, Lan et al. ont expérimenté avec un modèle nommé ALBERT (en anglais, *A Lite BERT*) afin de réduire le nombre de paramètres de BERT dans le but de mitiger le besoin de mémoire et augmenter la vitesse d'entraînement (Lan *et al.*, 2019). Les auteurs utilisent deux techniques.

Premièrement, ils factorisent le plongement d'entrée en le décomposant en deux matrices plus petites. Au lieu de projeter les vecteurs d'entrée directement dans un plongement de taille H, on les projette d'abord dans un espace d'intégration de dimension inférieure de taille E, pour ensuite les reprojeter dans le plongement. On réduit ainsi le nombre de paramètres du plongement d'entrée dans l'ordre de $\mathcal{O}(V \times H)$ à $\mathcal{O}(V \times E + E \times H)$.

Deuxièmement, ils partagent les paramètres entre les couches (en anglais, *parameter sharing*). Il existe plusieurs manières de partager des paramètres entre les couches. Par exemple, en partageant seulement les paramètres des couches *feed-forwards*, ou partageant seulement les paramètres du mécanisme d'attention (Lan *et al.*, 2019). Les auteurs ont observé de meilleurs résultats en partageant tous les paramètres entre les couches.

RoBERTa. Lui et al. ont expérimenté avec un modèle nommé RoBERTa (en anglais, *A Robustly Optimized BERT Training Approach*) en modifiant BERT de quatre façons notables : **(i)** entraîner le modèle plus longtemps, avec des lots plus importants, sur plus de données ; **(ii)** supprimer la tâche de prévision de la phrase suivante ; **(iii)** permettre de plus longues séquences d'entrée ; et **(iv)** permuter dynamiquement le masquage des jetons (Liu *et al.*, 2019).

CamemBERT. Afin de reproduire et de valider des résultats qui n'avaient été obtenus que pour l'anglais, Martin et al. tirent parti de corpus multilingues pour former un modèle de langue monolingue français. Le modèle se nomme CamemBERT. CamemBERT est très similaire à RoBERTa, la principale différence étant l'algorithme de masquage et la segmentation des mots avec SentencePiece (Martin *et al.*, 2019)

DeBERTa He at al proposent une nouvelle architecture de modèle DeBERTa (pour en anglais, *Decoding-enhanced BERT with untangled attention*) qui améliore les modèles BERT et RoBERTa en utilisant deux nouvelles techniques (He *et al.*,).

Premièrement, on introduit un mécanisme d'attention de débrouillage (pour en anglais, *untangled*), où chaque mot est représenté à l'aide de deux vecteurs qui encodent respectivement son contenu et sa position. Par exemple, soit les phrases X_1 et X_2 suivantes :

$$X_1 = \langle \text{le, ciel, est, bleu} \rangle ,$$

$$X_2 = \langle \text{le, ciel, est, beau, car, il, est, bleu} \rangle .$$

Les auteurs considèrent qu'il y a une plus grande dépendance entre *ciel* et *bleu* dans X_1 étant donnée une distance relative plus petite (He *et al.*,). Ainsi, le mécanisme d'attention est calculé à l'aide de matrices de débrouillage prenant en compte leur contenu et leurs positions relatives. Pour un jeton à une position i d'une séquence, les vecteurs H_i et $P_{i|j}$ représentent respectivement son contenu et sa position relative au jeton à la position j. Le calcul du score d'attention entre les jetons i et j peut être décomposé en quatre composantes se traduisant avec les équations suivantes :

$$\begin{aligned} \text{attention}_{i,j} &= \{H_i, P_{i|j}\} \times \{H_j, P_{j|i}\}^\top \\ &= H_i H_j^\top + H_i P_{j|i}^\top + P_{i|j} H_j^\top + P_{i|j} P_{j|i}^\top \end{aligned} \tag{4.14}$$

Autrement dit, à l'aide de matrices de débrouillage prenant en compte le contenu et les positions

relatives d'une paire de mots, le poids d'attention peut être calculé comme la somme de quatre scores d'attention, c'est-à-dire les scores de *contenu-à-contenu*, de *contenu-à-position*, de *position-à-contenu* et de *position-à-position*.

Deuxièmement, au lieu d'utiliser l'encodage positionnel comme moyen de prétraitement de la séquence d'entrée, celui-ci est ajouté à tous les blocs d'encodeur avant chaque couche *softmax*. Les auteurs affirment que de cette façon, DeBERTa capture les positions relatives dans tous les blocs du transformeur et utilise les positions absolues comme informations complémentaires lors du décodage des mots masqués (He *et al.*,).

Bref, comparativement à BERT, AlBERT est une version qui réduit la complexité spatiale, RoBERTa, une version optimisée, CamemBERT, une version française optimisée et DeBERTa une version optimisée qui prend en compte les positions relatives des mots.

4.3 GPT

Le modèle GPT (en anglais, pour *Generative Pre-trained Transformer*) est un modèle de langue de type transformeur multicouche bâti sur une pile de décodeurs (Radford *et al.*, 2018). Les auteurs utilisent une variante du décodeur où celui-ci crée lui-même ses valeurs K et Q au lieu de les recevoir de l'encodeur.

Outre le fait que le transformeur GPT emploie des décodeurs, le transformeur GPT diffère aussi de BERT puisqu'il génère un jeton à la fois. Après la production d'un jeton, ce même jeton est ajouté à la séquence d'entrées. Et cette nouvelle séquence devient l'entrée pour la prochaine étape.

Le modèle se décline en quatre versions. Leurs différences se situent surtout en termes de nombre de paramètres et non pas au niveau de l'architecture.

- GPT-1 (2018) contient un nombre de paramètres dans le même ordre que le transformeur original, c'est-à-dire autour de 110 millions de paramètres,
- GPT-2 (2019) porte le nombre de paramètres à 1,5 milliard,
- GPT-3 (2020), augmente le nombre de paramètres à 175 milliards,
- GPT-4 (2023), le nombre de paramètres atteint 1 trilliard, ce qui en fait un des plus gros

réseaux de neurones dans le domaine du TAL à ce jour (Floridi et Chiriatti, 2023).

4.3.1 Préentraînement

Le jeu de données BooksCorpus est notamment utilisé pour préentraîner le modèle. Il contient plus de 7000 livres de divers genres, par exemple d'aventure, de fantaisie et de romance. Il contient aussi de longues portions de texte contiguës, ce qui permet au modèle génératif d'apprendre à se conditionner sur des informations à longue portée. La figure 4.9 illustre comment on peut générer des entrées pour le modèle GPT.

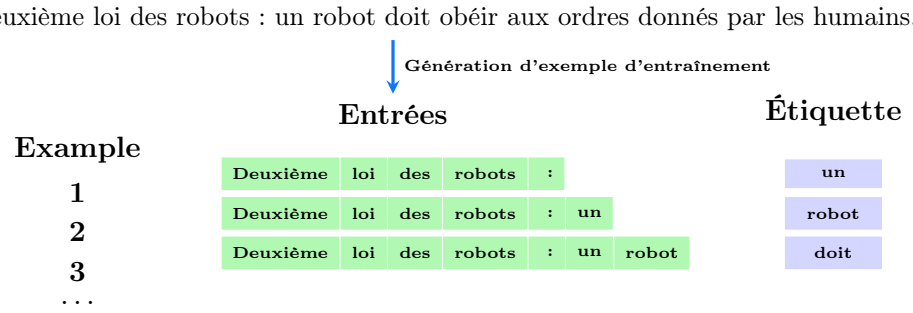

FIGURE 4.9 – Adapté de (Jalamar, 2020a). Il est possible de générer une quantité illimitée d'exemples de formation depuis un corpus de texte, car il suffit de prendre un mot ainsi que le contexte le précédant.

Étant donné un corpus de jetons $B = b_1, \ldots, b_n$, Radford et al. forment un modèle qui a pour objectif de maximiser la probabilité d'un jeton b par rapport à sa fenêtre de contexte de taille k, c'est-à-dire les k jetons observés auparavant (Radford *et al.*, 2018). Ce modèle utilise le mécanisme d'attention multitêtes sur les jetons d'entrée, suivie d'une couche d'encodage positionnel afin de produire une distribution de sortie sur les jetons cibles, où n désigne le nombre de couches, ℓ désigne la profondeur d'une couche, E désigne le plongement des jetons, W_p désigne l'encodage positionnel et Θ désigne les paramètres du modèle. La probabilité conditionnelle p est calculée à l'aide d'une couche *softmax*. L'entrée de la couche *softmax* est une projection linéaire du plongement avec la dernière couche d'états cachée de la pile de décodeurs.

Cela se traduit par les équations :

$$\begin{aligned} h_0 &= E + W_p, \\ h_\ell &= \text{decoder}(h_{\ell-1}) \text{ pour tout } \ell \in [1, n], \\ p(b) &= \text{softmax}(h_n E^\mathsf{T}), \\ J &= -\sum \log p(b \mid b_{-k}, \ldots, b_{-1}, \Theta). \end{aligned} \tag{4.15}$$

4.3.2 Affinement

Une fois le modèle préformé, on évalue et affine celui-ci sur différentes tâches du TAL.

Classification. Le *Corpus of Linguistic Acceptability* (CoLA) contient des jugements d'experts sur le caractère grammatical ou non d'une phrase et teste le biais linguistique des modèles entraînés. Le *Stanford Sentiment Treebank* (SST-2), en revanche, concerne une tâche de classification binaire standard.

Implication textuelle. La tâche d'implication textuelle consiste à lire une paire de phrases et à juger de la relation entre elles à partir d'une implication, d'une contradiction ou d'une neutralité. La tâche reste difficile en raison de la présence d'une grande variété de phénomènes tels que l'implication lexicale, la coréférence et l'ambiguïté lexicale et syntaxique.

Similarité sémantique. Les tâches de similarité sémantique consistent à prédire si deux phrases sont sémantiquement équivalentes ou non. Les défis consistent à reconnaître la reformulation des concepts, à comprendre la négation et à gérer l'ambiguïté syntaxique. Radford et al. utilisent trois ensembles de données pour cette tâche soit : (**i**) le corpus *Microsoft Paraphrase*, (**ii**) l'ensemble de données *Quora Question Pairs* et (**iii**) l'étalon de référence (en anglais, *benchmark*) *Semantic Textual Similarity*.

Choix multiples. La tâche de réponse aux questions en est une qui nécessite des aspects de raisonnement pouvant s'étaler sur une ou plusieurs phrases. Radford et al. utilisent l'ensemble de données RACE qui est composé de réponse à des questions associées à des examens du collège et du lycée. Il a été démontré que ce corpus contient davantage de questions de type raisonnement que d'autres ensembles de données comme CNN ou SQuaD et qu'il fournit donc une meilleure

évaluation du modèle quant à sa gestion des dépendances contextuelles à longue portée (Radford *et al.*, 2018). Le modèle est aussi évalué sur l'ensemble de données de test *Story Cloze Test*, qui consiste à sélectionner, parmi deux options, la fin appropriée à des histoires contenant plusieurs phrases.

La figure 4.10 illustre comment le modèle GPT peut être utilisé sur un éventail de tâches dans une optique d'affinage du modèle.

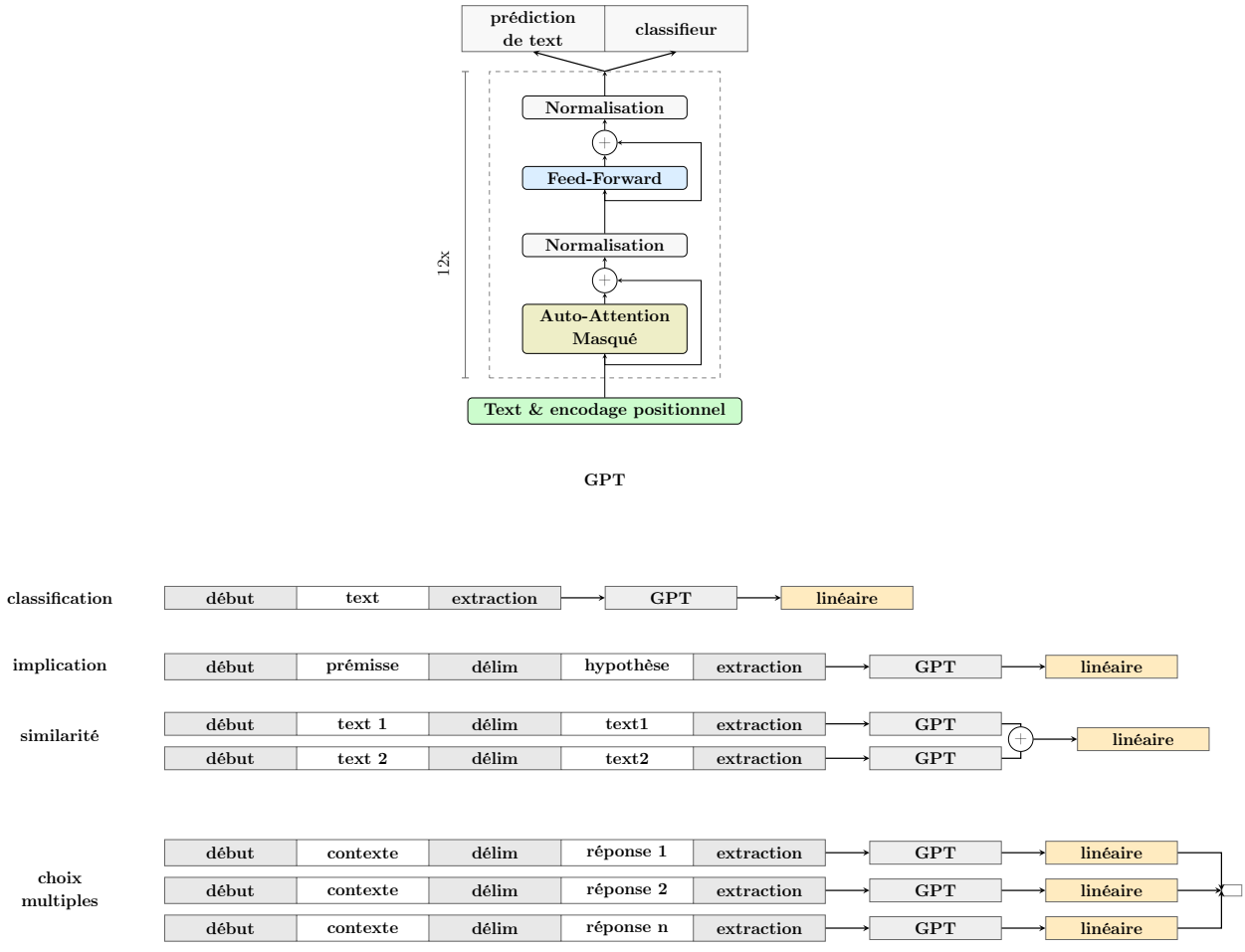

FIGURE 4.10 – Adapté de (Radford *et al.*, 2018). Lorsque préformé, le transformeur GPT peut être utilisé de multiples façons afin de résoudre plusieurs types de tâche du TAL, dont les tâches de classification, d'implication textuelle, de similarité sémantique et de réponse à choix multiples.

Sur ces tâches, le modèle surpasse les meilleurs résultats obtenus auparavant par d'autres modèles et avec des marges significatives. Cela démontre la capacité de GPT à gérer efficacement les contextes à longue portée (Radford *et al.*, 2018). Au fil des versions, les modèles deviennent de plus en plus

gros et d'autres tâches y sont introduites. Par exemple, GPT-4 est en mesure de *voir*, c'est-à-dire que le modèle reconnaît et décrit des images imbriquées dans le texte.

4.3.3 ChatGPT

ChatGPT est un chatbot construit à l'aide du transformeur GPT-4 et affiné avec des techniques d'apprentissage supervisé et par renforcement. Dans le cas de l'apprentissage supervisé, le modèle prend en entrée des conversations dans lesquelles les auteurs jouent les deux rôles, c'est-à-dire l'utilisateur et l'assistant IA. Dans l'étape de renforcement, les auteurs ont d'abord classé en ordre de rang les réponses que le modèle avait créées lors de conversations précédentes. Ces classements sont utilisés afin de récompenser le modèle selon une politique d'optimisation (OpenAi, 2022).

ChatGPT est capable de faire preuve d'esprit critique et de générer un texte très réaliste avec une entrée minimale, ce qui en fait une menace potentielle pour l'intégrité des examens en ligne, en particulier dans les établissements d'enseignement supérieur où ces examens sont de plus en plus répandus (Susnjak, 2022).

4.4 XLNet

XLNet est essentiellement une pile de décodeurs, telle que GPT-2, utilisant une technique qui encode les mots comme le fait BERT (voir section 4.2) et le transformeur original voir section 4.1 XLNel est un PML (en anglais, *Permutation Model Language*) qui capture le contexte bidirectionnel sur toutes les permutations possibles de mots dans une phrase. Ainsi, XLNet maximise la vraisemblance logarithmique sur toutes les permutations possibles de la séquence d'entrée, c'est-à-dire que chaque position apprend à utiliser les informations contextuelles de toutes les positions capturant ainsi le contexte bidirectionnel. Aucun jeton de masquage n'est nécessaire.

4.4.1 Motivation

Il existe deux principales limitations avec le modèle BERT : (**i**) dans des applications réelles, il n'y a pas d'entrées masquées et (**ii**) BERT néglige les dépendances entre les positions masquées.

Par exemple, considérons la phrase $P = \langle$est, une, ville, New, York$\rangle$ suivant une procédure de masquage des jetons `New` et `York`. Alors, l'objectif de modélisation de BERT se traduit par :

$$\log p(\text{New} \mid \text{est une ville}) + \log p(\text{York} \mid \text{est une ville}) \tag{4.16}$$

Dans l'objectif ci-dessus, il n'y a pas de dépendance entre `New` et `York`.

Étant un modèle autorégressif, XLNet prédit dans l'ordre de la séquence, c'est-à-dire qu'il prévoit d'abord le jeton `New`, puis le jeton `York`. Cela se traduit par l'équation :

$$\log p(\text{New} \mid \text{est une ville}) + \log p(\text{York} \mid \text{est une ville New}) \tag{4.17}$$

XLNet est donc un modèle autorégressif qui encode et décode les données sans les masquées.

4.4.2 Attention masquée à double-flux

Afin de garder une distribution uniforme des données, le modèle doit *masquer* les jetons sans le jeton `[MASK]` tout en encodant la séquence. Cela est d'autant plus complexe puisqu'un encodage positionnel (voir section 4.1.1) est ajouté au plongement d'entrée. Pour ce faire, le modèle utilise une attention masquée à double-flux, où un flux de contenu h encode la séquence et un flux de requête g empêche (*masque*) un jeton à prédire d'avoir accès aux informations le concernant.

Les décodeurs sont mis à jour selon les équations suivantes :

$$\begin{aligned}
\hat{h}_{z_t}^{\ell} &= \text{Norm}\left(h_{z_t}^{\ell-1} + \text{Att}(\text{Q} = h_{z_t}^{\ell-1}, \text{K} = h^{\ell-1}, \text{V} = h_{z<t}^{\ell-1})\right), \\
h_{z_t}^{\ell} &= \text{Norm}\left(\hat{h}_{z_t}^{\ell} + \text{FF}(\hat{h}^{\ell})\right), \\
\hat{g}_{z_t}^{\ell} &= \text{Norm}\left(g_{z_t}^{\ell-1} + \text{Att}(\text{Q} = g_{z_t}^{\ell-1}, \text{K} = h^{\ell-1}, \text{V} = h_{z<t}^{\ell-1})\right), \\
g_{z_t}^{\ell} &= \text{Norm}\left(\hat{g}_{z_t}^{\ell} + \text{FF}(\hat{g}^{\ell})\right),
\end{aligned} \tag{4.18}$$

avec

- Norm : fonction de normalisation,
- Att : mécanisme d'attention masquée,

— FF : couche linéaire (*feed-forward*),

— z : séquence selon l'ordre de factorisation,

— t : indice du mot traité,

— h_z^0 : plongement de la séquence d'entrée factorisée,

— g_z^0 : vecteurs de poids entraînables.

La figure 4.11 illustre le mécanisme d'attention masquée à double-flux selon un ordre de factorisation 3→2→4→1 :

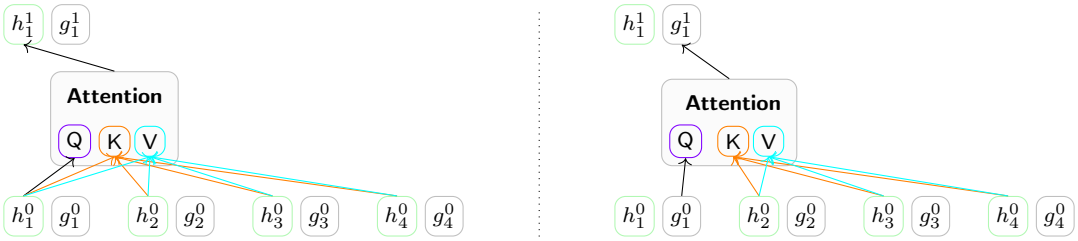

FIGURE 4.11 – Adapté de (Yang *et al.*, 2019). Pour le flux de contenu (à gauche), le mécanisme d'attention masquée à double-flux suit la même méthode que le transformeur original. Le mécanisme d'attention masquée pour le flux de requête (à droite) empêche le modèle de porter attention au jeton à prévoir.

L'encodage positionnel (voir 4.1.1) ne dépend que des positions réelles de la séquence d'origine. La mise à jour des décodeurs est donc indépendante des permutations une fois $\tilde{h}(\ell)$ obtenues. Ultimement, le modèle apprend à utiliser la mémoire sur tous les ordres de factorisation du dernier segment permuté.

4.4.3 Permutations

Étant donnée que XLNet permute la séquence d'entrée précédent le jeton à prévoir, pour la prévision du jeton `York` de l'exemple précédant, cela se traduit par l'objectif :

$$\begin{aligned} X &= \text{Perm}(\text{est une ville New}), \\ &\sum_{x \in X} \log p(\text{York} \mid x). \end{aligned} \tag{4.19}$$

où

— Perm : fonction retournant un ensemble de séquence comprenant toutes les permutations

possibles d'une séquence.

Afin de maximiser cet objectif, on distingue un masque d'attention pour chacun des flux. Par exemple, supposons une entrée $P = \langle le, ciel, est, bleu \rangle$ où la séquence d'entrée selon un ordre de factorisation 3→2→4→1 devient $z = \langle est, ciel, bleu, le \rangle$. Dans ce cas, le masque d'attention pour le flux h est construit de façon à ce que :

— le jeton `le` porte attention à tous les jetons (`le, ciel, est, bleu`),

— le jeton `ciel` porte attention à deux jetons (`ciel, est`),

— le jeton `est` porte attention à un jeton (`est`),

— le jeton `bleu` porte attention à trois jetons (`ciel, est, bleu`).

Le masque pour le flux g est construit de la même façon avec l'unique différence que les jetons ne peuvent se porter attention à eux même.

La figure 4.12 illustre le procédé de masquage du modèle XLNet.

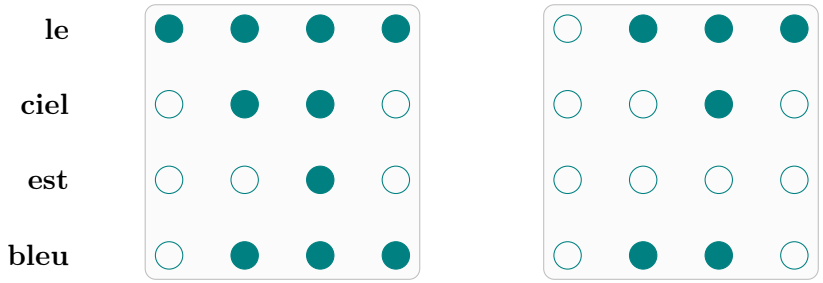

FIGURE 4.12 – Adapté de (Yang *et al.*, 2019). Le masque d'attention du flux de contenu h (à gauche), et le masque d'attention de requête g (à droite) sont représentés par des matrices dans lesquelles les points blancs ont une valeur numérique égale à $-\infty$. Le masque d'attention de h, est un masque d'attention tel que pour le transformeur orignial, mais où les lignes sont permutées selon l'ordre de factorisation. Le masque d'attention de g suit la même logique à l'exeption qu'un jeton ne peut se porter attention à lui même.

Ainsi, le modèle apprend à maximiser l'objectif :

$$p(X_{z_t} \mid x_{z<t}) = \frac{\exp\left(e(x)^\top g_{z_t}^M\right)}{\sum_{x'}\left(\exp e(x')^\top g_{z_t}^M\right)} \tag{4.20}$$

XLNet surpasse BERT sur 20 tâches dont la réponse aux questions et l'analyse des sentiments (Yang

et al., 2019). Ce modèle prend plus de temps à former et nécessite davantage de mémoire que GPT étant donné le nombre de paramètres qu'implique son mécanisme d'attention.

4.5 Dictionnaire inversé basé sur transformeurs

4.5.1 BERT NRD

Yan et al. ont expérimenté avec un modèle basé sur BERT. Comme pour la tâche originale de BERT, où le réseau doit faire des prévisions sur les masques, leur modèle a pour objectif d'associer le mot cible w à partir des k masques en fonction de la définition P (Yan *et al.*, 2020).

La séquence d'entrée prend alors la forme
`[CLS] + [MASK] * k + [SEP] + [sequence de sous-mots de la definition P] + [SEP].`

On définit le modèle comme suit :

- $|V|$: le nombre de jetons,
- h_0 : l'entrée du modèle,
- Norm : la couche de normalisation,
- Att : le mécanisme d'attention multitêtes,
- ℓ : la profondeur d'une couche,
- H_k^L : les états cachés pour les k jetons masqués de la sortie de BERT,
- MLM : le modèle de langage masqué,
- $score_b$: la distribution des scores des jetons pour les k positions,
- ϕ : contient trois couches, une couche de projection linéaire, suivie d'une couche d'activation, puis d'une autre couche de projection linéaire.

$$\begin{aligned} \hat{h^\ell} &= \text{Norm}(h^{\ell-1} + \text{Att}(h^{\ell-1})), \\ h^\ell &= \text{Norm}(\hat{h^\ell} + \phi(\hat{h^\ell})), \\ \text{score}_b &= \text{MLM}(H_k^L). \end{aligned} \tag{4.21}$$

Score d'un mot. Après avoir atteint score_b, on doit retrouver le score pour le mot cible, puisque score_b représente le score pour les sous-mots. Cependant, il existe $|V|^k$ combinaisons de sous-mots,

ce qui rend difficile la représentation des mots par concaténation de sous-mots. Le nombre de mots uniques dans le dictionnaire d'une langue est limité. Par conséquent, il est possible de se soucier uniquement des séquences de sous-mots qui peuvent former un mot valide. On répertorie d'abord tous les mots valides et on trouve leur séquence de sous-mots. Ensuite, on calcule le score pour un mot valide avec sa séquence de sous-mots $[b_1, \ldots, b_k]$. Puis, le réseau est entraîné afin de minimiser l'entropie croisée entre la prévision du modèle et le mot cible.

Cela se traduit par les équations :

$$\begin{aligned} \text{score}_w &= \sum_i^k \text{score}_{b_i}, \\ J_w &= -\sum_i^{|V|} w_i \log(\text{score}_{w_i}). \end{aligned} \tag{4.22}$$

La figure 4.13 illustre comment on peut utiliser BERT et la tâche de prévision de jetons afin de résoudre le problème du dictionnaire inversé.

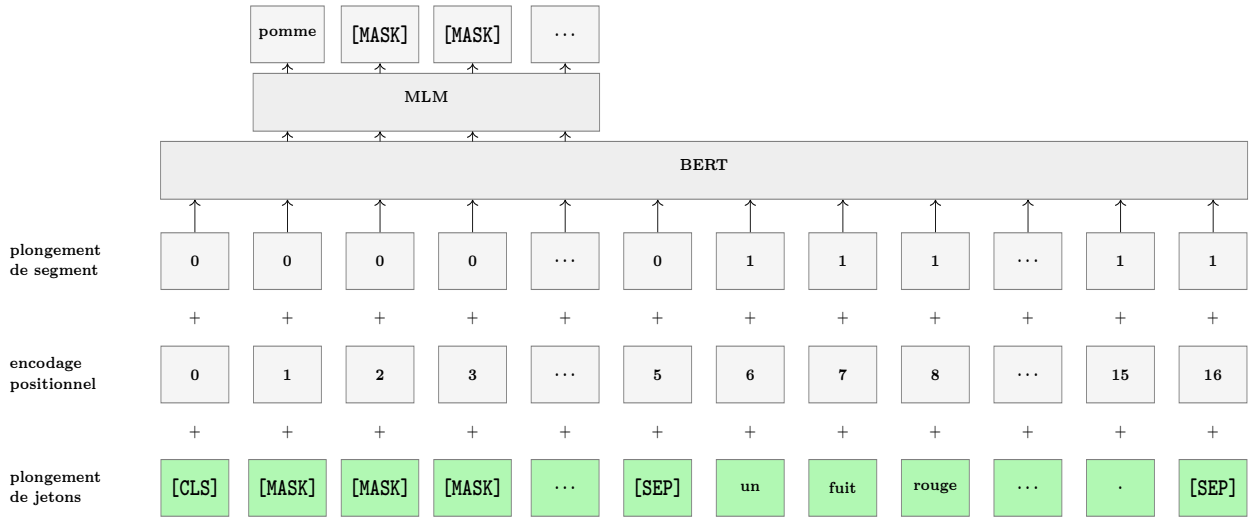

FIGURE 4.13 – Adapté de (Yan *et al.*, 2020). Comme dans la tâche originale de prévision des masques, le modèle tente de prévoir les masques pour une définition donnée. Puisque BERT prévoit des sous-mots, durant l'inférence, on doit reconstruire un mot à partir des sous-mots.

4.5.2 Augmentation des données

Les définitions du dictionnaire ne correspondent pas nécessairement au type de requêtes qu'un utilisateur pourrait saisir. Newman et Aydin appliquent l'augmentation des données sous la forme de requêtes de synonymes, d'antonymes et de parties du discours. Ils proposent un modèle basé sur BART (Lewis *et al.*, 2020) et un modèle basé sur T5 (Xue *et al.*, 2021) et qui sont entraînés sur des données augmentées. Ils constatent que la précision de leurs modèles se sont améliorée (Newman et Aydin, 2022).

4.5.3 Plongement de mot partagé

Un plongement de mot a la même signification de ses définitions, bien que leurs formes diffèrent. Chen et Zhao proposent un modèle basé sur BERT dans lequel les mots et les définitions partagent un espace vectoriel (Chen et Zhao, 2022). Pour ce faire, le modèle est formé sur deux tâches, soit (**i**) la tâche du dictionnaire inversé et (**ii**) la remodélisation de la définition selon le mot d'entrée (voir section 4.5.4).

La figure 4.14 illustre comment les mots et les definitions peuvent partager un même espace vectoriel.

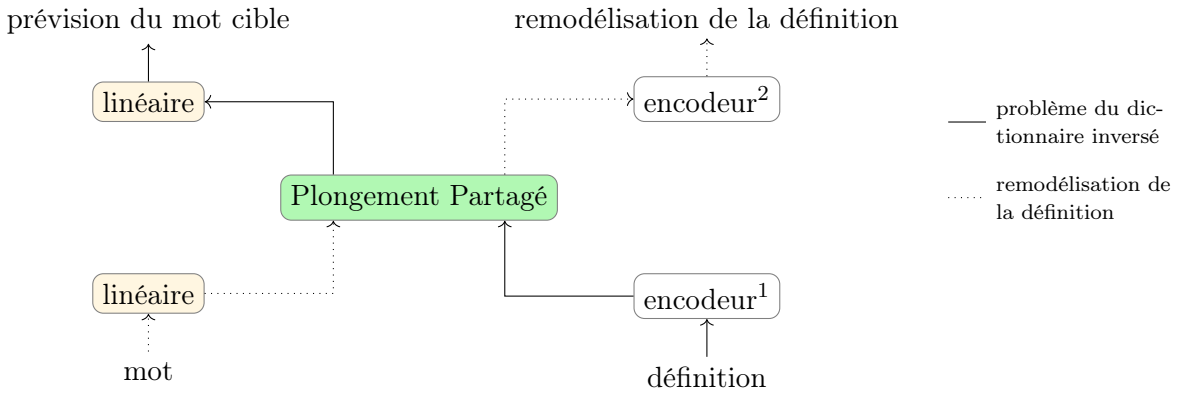

FIGURE 4.14 – Adapté de (Chen et Zhao, 2022). Le modèle partage un même espace de plongement qui est utilisé afin de résoudre deux tâches connexes, c'est-à-dire la remodélisation d'une définition et le problème du dictionnaire inversé. Le plongement partagé entre BERT et une couche linéaire est connecté entre deux blocs d'encodeur.

Ainsi, la représentation partagée peut être vue comme un auto-encodage du sens d'un mot et de sa

définition. Présentement, le modèle est limité à faire des prévisions en anglais.

4.5.4 Comparaison des dictionnaires et des plongements de mots

La comparaison des dictionnaires et des incorporations de mots (en anglais, *Comparing Dictionaries and Word Embeddings* (CODWOE) est une tâche visant à comparer la remodélisation des définitions et la *tâche de dictionnaire inversé*. La tâche de remodélisation de la définition consiste à reconstruire la définition pour un mot donnée. Afin de résoudre ces tâches, on fournit des ensembles de données dans cinq langues différentes. Li et al. ont pré-entraîné un modèle multilingue basé sur DeBERTa (voir section 4.2.8) et initialisé de manière aléatoire sur chaque ensemble de données (Li *et al.*, 2022). Ensuite, on évalue le modèle en comparant le plongement de sortie avec un plongement de référence suivant trois métriques : la similarité cosinus, l'erreur quadratique moyenne et le rang cosinus, c'est-à-dire combien d'autres items de test ont une similarité cosinus plus élevée.

CHAPITRE 5
EXPÉRIMENTATION

Rappelons que pour résoudre le problème du dictionnaire inversé, il faut trouver le mot cible w à partir de sa définition $P = \langle w_1, w_2, \ldots, w_n \rangle$. Dans une étude précédente, il a été démontré que l'on pouvait modéliser BERT et résoudre le problème du dictionnaire inversé de la même manière que la tâche originale de prévisions des jetons masqués (Yan *et al.*, 2020). La différence de notre approche réside dans l'emploi de différents transformeurs comme plongement à la base de nos modèles. À partir de modèles reposant sur les transformeurs XLNet, BERT et GPT-2, nous affinons ceux-ci sur la tâche du dictionnaire inversé afin de les comparer. À titre applicatif, nous utilisons aussi les modèles entraînés afin d'évaluer des dictionnaires provenant du *jeu du dictionnaire*. Nous introduisons d'abord les transformeurs, puis présentons la méthode que nous avons utilisée pour incorporer les transformeurs afin de résoudre le problème du dictionnaire inversé.

L'architecture originale d'un transformeur comprend une pile d'encodeurs et une pile de décodeurs de même taille. Pour le premier encodeur, l'entrée est le plongement de mots, mais dans les couches subséquentes, l'entrée est la sortie de l'encodeur qui se trouve directement au-dessous. Les sorties de la couche d'auto-attention sont transmises à une couche non linéaire puis vers l'encodeur suivant. La sortie du dernier encodeur est transmise aux différents décodeurs inférieurs et ceux-ci font remonter leurs résultats de décodage comme l'ont fait les encodeurs. Le processus est répété jusqu'à ce qu'un symbole spécial soit atteint par une dernière couche *softmax* connectée à la sortie de la pile de décodeurs indiquant que le transformeur a terminé le traitement.

BERT apprend à partir de représentations bidirectionnelles de texte non étiqueté en conditionnant conjointement les contextes gauche et droit dans toutes les couches. En conséquence, BERT peut être affiné en ajoutant une couche de sortie afin de créer des modèles destinés à résoudre un éventail de tâches.

GPT-2 est un modèle de langue transformeur multicouche composé d'une pile de 12 décodeurs. Les décodeurs sont des variantes où ceux-ci créent eux-mêmes leurs valeurs $\mathbf{K}$ et $\mathbf{Q}$ au lieu de les recevoir de l'encodeur.

XLNet est un modèle *autorégressif* tel que GPT et bidirectionnel tel que BERT. Cependant, au lieu d'utiliser un ordre de jetons fixe vers l'avant ou vers l'arrière, XLNet maximise la vraisemblance logarithmique d'une séquence en considérant toutes les permutations possibles des jetons. Grâce à l'opération de permutation, le contexte de chaque position peut être constitué de jetons situés à gauche et à droite (Yang *et al.*, 2019).

5.1 Données

L'ensemble de données de base que nous utilisons a été assemblé par Hill et al. (Hill *et al.*, 2016). Les définitions sont extraites de cinq ressources électroniques : *WordNet*, *The American Heritage Dictionary*, *The Collaborative International Dictionary of English*, *Wiktionary* et *Merriam Webster's*. La plupart des mots dans les données d'entraînement ont plusieurs définitions. Pour chaque mot w avec des définitions $\{P_1, \ldots, P_n\}$ les auteurs incluent toutes les paires $(w, P_1) \ldots (w, P_n)$ dans l'échantillon de données. Pour permettre aux modèles d'accéder à plus de connaissances factuelles que celles qui pourraient être présentes dans un dictionnaire (par exemple, des informations sur des entités, des lieux ou des personnes spécifiques), les auteurs complètent les données avec des informations extraites de Wikipédia. Les phrases du premier paragraphe de l'article sont traitées comme si elles étaient des définitions indépendantes de ce mot. Lorsqu'un mot de Wikipédia apparaît également dans un ou plusieurs des cinq dictionnaires initiaux, des pseudo-définitions pour ce mot sont ajoutées à l'ensemble de données (Hill *et al.*, 2016). L'ensemble de données contient 108 420 mots et 900 000 paires mot-définitions. Il y trois jeux de test comprenant : **(i)** un ensemble de définitions vues, qui contient 500 paires mot-définitions présentes dans l'ensemble de données d'entraînement. **(ii)** un ensemble de définitions non vues, qui contient également 500 paires mot-définitions dont les paires mots-définitions ont été exclues de l'ensemble de formation ; et **(iii)** un ensemble de description, composé de 200 paires mot-définitions écrites par l'humain.

Étant donnée qu'il s'agit d'un ensemble de données de référence utilisé dans plusieurs autres études (Hill *et al.*, 2016; Morinaga et Yamaguchi, 2018; Zhang *et al.*, 2020; Chen et Su, 2021), nous considérons cet ensemble comme étant complet et sans erreurs.

5.2 Prétraitement

Nous avons d'abord utilisé la librairie **spaCy** (Honnibal et Montani, 2017) afin d'ajouter à l'ensemble de données le POS correspondant à chacun des mots cibles. La librairie compte 16 POS distincts, soit : `ADJ,ADP, ADV, AUX, CONJ, CCONJ, DET, INTJ, NOUN, NUM, PART, PRON, PROPN, PUNCT, SCONJ, SYM, VERB, X, SPACE`.

Aussi, à partir de l'ensemble de données que nous utilisons, peu de définitions excèdent 50 jetons après avoir été segmentées avec l'algorithme WordPiece (Wu *et al.*, 2016). Par conséquent, nous avons limité la taille du modèle à des phrases d'une longueur maximale fixée à 50 jetons. Les paires mot-définitions ayant des définitions excédant 50 jetons sont exclues des données.

5.3 Phase d'apprentissage

Nous utilisons les modèles préentraînés des transformeurs BERT, GPT-2 et XLNet et les connectons à une couche de *dropout* suivi d'une couche *softmax* qui calcule les probabilités que $w \in V$ soit associé à une définition P. Avec BERT, nous utilisons la sortie du jeton `[CLS]`, puisque ce jeton renferme déjà les informations pour une tâche de classification. Avec GPT-2, nous utilisons la sortie du dernier jeton du dernier décodeur, puisque ce jeton aura pris connaissance des autres jetons étant donné la nature du mécanisme d'attention du décodeur. Avec XLNet, nous utilisons la moyenne des états cachés de tous les jetons de la couche de sortie en excluant les jetons associés au remplissage (en anglais, *padding*).

La figure 5.1 illustre comment nous utilisons les sorties des transformeurs à l'étude afin de représenter la phrase d'entrée.

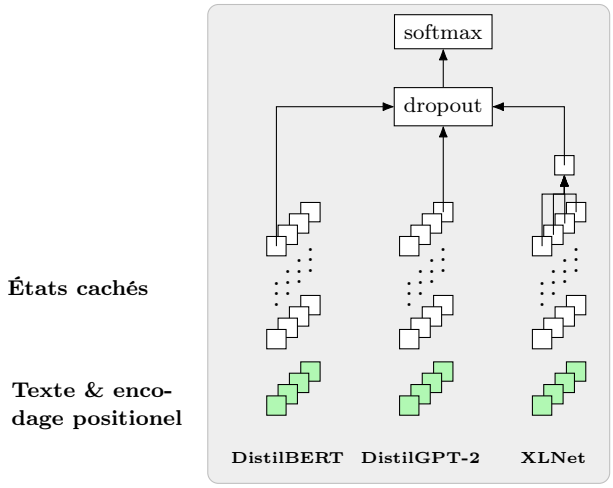

FIGURE 5.1 – Sorties des transformeurs pour les architectures DistilBERT, DistilGPT-2 et XLNet. En considérant le dernier état caché d'un transformateur pour une phrase d'entrée, nous utilisons le jeton `[CLS]` pour DistilBERT, le dernier jeton pour DistilGPT-2 et une moyenne des jetons pour XLNet.

Le modèle minimise la fonction de coût obtenue par entropie croisée. Nous ajoutons une tâche auxiliaire suivant la même configuration que la tâche précédente afin de prédire le POS du mot cible. Nous utilisons les versions distillées des transformeurs BERT et GPT-2 qui ont été implémentés par Hugging Face et qui sont appelés DistilBERT et DistilGPT-2 (Face, 2022). Les versions distillées retiennent la majorité des connaissances des versions originales tout en diminuant le nombre de paramètres (Sanh *et al.*, 2019). De plus, ces modèles sont implémentés selon des interfaces qui se connectent aisément aux librairies *Tensorflow* et *PyTorch*.

Les modèles ont été entraînés à l'aide d'un GPU NVIDIA V100 en se connectant au serveur de l'*Alliance de Recherche Numérique du Canada* (Baldwin, 2012) sur le noeud *Béluga*[1]. Le nombre d'époques est fixé à 200 pour les modèles DistilBERT et DistilGPT-2 ainsi qu'à 130 pour le modèle XLNet. Cela représente une semaine d'entraînement soit le maximum permis par le noeud *Béluga*. Il est toutefois possible de poursuivre l'entraînement en rechargeant le modèle, mais l'optimiseur doit repasser par la phase de réchauffement d'autant plus que la fonction de coût ne diminue presque plus.

La figure 5.2 illustre la fonction de coût et la précision du modèle durant la phase d'apprentissage.

1. https ://docs.alliancecan.ca/wiki/B%C3%A9luga/en

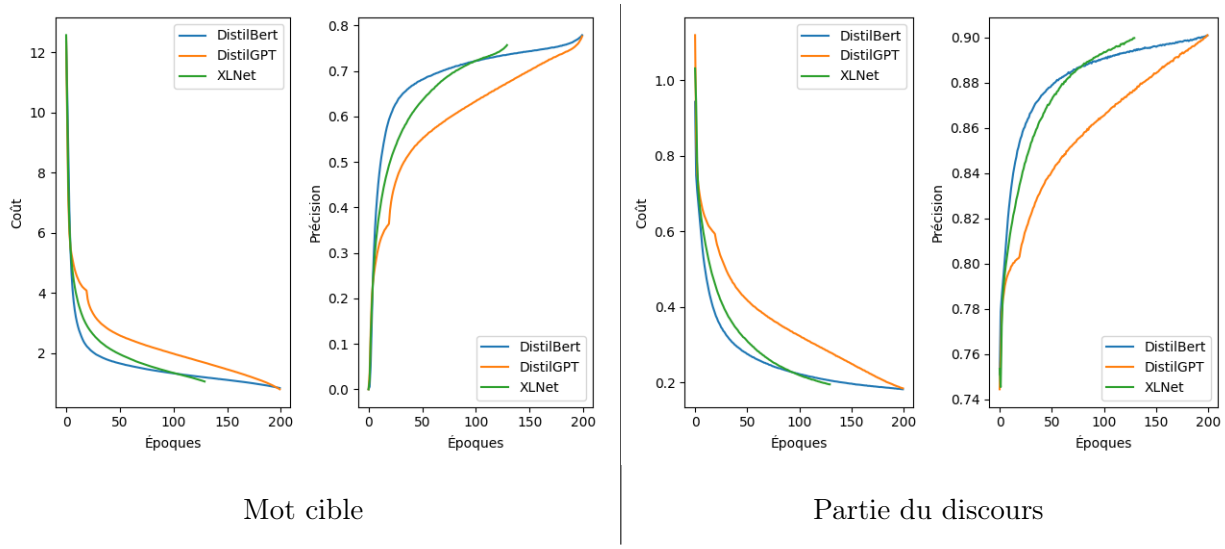

FIGURE 5.2 – Nos modèles basés sur DistilBERT, DistilGPT-2 et XLNet sont entrainés durant 200, 200 et 130 époques respectivement. Les modèles sont formés de façon à prévoir, pour une définition donnée, le mot cible (à gauche) et le POS (à droite) associé. La fonction de coût repose sur l'entropie croisée. La précision du modèle est représentée sur une échelle de 0 à 1. La précison est calculée avec le nombre de fois que le modèle prévoit avec succès un mot cible ou un POS pour toutes définitions tirées des données d'entraînement, divisé par le nombre total de définitions incluses dans les données d'entraînement. On remarque que le modèle basé sur XLNet converge un peu plus vite. Cela est peut-être dû au fait que ce n'est pas un modèle qui a été distillé.

Paramètres. La normalisation par lots (en anglais, *batch normalisation*) est une technique de régularisation utilisée dans les réseaux de neurones qui peut réduire les risques de surapprentissage (Ioffe et Szegedy, 2015). Le *dropout* est une autre technique pour réduire le surapprentissage en supprimant temporairement des neurones selon une distribution de Bernoulli (Srivastava *et al.*, 2014). On peut voir cette technique comme un moyen forçant un entraînement aléatoire de sous-réseaux du réseau. Des études ont démontré qu'il est parfois préférable de ne pas combiner les deux techniques (Li *et al.*, 2019; Garbin *et al.*, 2020). Aussi, certaines études indiquent que le *dropout* peut donner de meilleurs résultats par rapport à la normalisation par lots (Park et Kwak, 2016; Garbin *et al.*, 2020).

Une couche de *dropout* est donc ajoutée à la sortie du transformeur. Nous émettons l'hypothèse que cela force le réseau à considérer différents liens sémantiques puisque certains neurones vont être désactivés aléatoirement à chaque pas d'apprentissage.

Le papier original qui a introduit la notion de *dropout* propose un taux arbitraire de 0.5 (Srivastava *et al.*, 2014). Nous avons obtenu de meilleurs résultats en fixant à 0.4 le taux de *dropout*.

Nous utilisons un optimiseur AdamW tel décrit à la section 4.2.4 se référant au transformeur BERT. Généralement, pour l'optimiseur Adam, on fixe un taux d'apprentissage à 0.001 (Wilson *et al.*, 2017). Cependant, nous avons observé que ce taux faisait diverger subitement (en anglais, *catastrophic forgettings*) le réseau de façon similairement rapportée dans une autre étude (Kirkpatrick *et al.*, 2017). Nous avons donc initialisé à 0.0001 le taux d'apprentissage comme le recommandent les auteurs de BERT pour les tâches d'affinages (Devlin *et al.*, 2019).

5.4 Résultats pour le problème du dictionnaire inversé

Nous avons évalué notre modèle sur les données collectées par Hill et al. selon trois métriques : le rang médian des mots cibles, le taux d'occurrence des mots cibles (précision) dans le top 1/10/100, et la variance du rang du mot cible (Hill *et al.*, 2016). Ces métriques sont également rapportées dans plusieurs autres études (Hill *et al.*, 2016; Morinaga et Yamaguchi, 2018; Zhang *et al.*, 2020; Chen et Su, 2021). De plus, puisqu'il est impossible d'exclure une définition de la base de données du modèle OneLook, l'évaluation de sa précision par rapport à l'ensemble de données de test non vues est notée comme étant sans objet.

Modèle	Auteur	Definition Vue			Definition Non vue			Description		
OneLook	Thesaurus	**0**	.66/.94/.95	200	-	-	-	5.5	.33/.54/.76	332
BoW	Hill	172	.03/.16/.43	414	248	.03/.13/.39	424	22	.13/.41/.69	308
RNN	Hill	134	.03/.16/.44	375	171	.03/.15/.42	404	17	.14/.40/.73	274
BiLSTM	Hill	25	.18/.39/.63	363	101	.07/.24/.49	401	5	.25/.60/.83	214
MS-LSTM	Kartsaklis	**0**	**.92**/.98/.99	65	276	.03/.14/.37	426	1000	.01/.04/.18	404
Multi-channel	Yamaguchi	16	.20/.44/.71	200	54	.09/.29/.58	**358**	2	.32/.64/.88	203
BERT(MLM)	Yan	**0**	.57/.86/.92	240	18	.20/**.46**/.64	418	**1**	36/**.66**/.94	**94**
Unified	Chen	-	-	-	18	.13/.39/**.81**	386	4	.22/.64/**.97**	183
DistilBERT(CLS)	Nous	**0**	.77/**.99**/**1.0**	**3**	**15**	**.24**/.45/.64	475	4	.31/.60/.80	397
DistilGPT-2	Nous	**0**	.79/**.99**/**1.0**	**3**	41	.18/.40/.57	488	4	.31/.57/.78	414
XLNet	Nous	**0**	.77/**.99**/**1.0**	**3**	**15**	**.23**/**.46**/.64	469	3	**.38**/.65/.85	364
			rang médian			précision@1/10/100			variance du rang	

TABLE 5.1 – Le tableau contient les résultats de notre expérimentation pour les transformeurs DistilBERT(CLS), DistilGPT-2 et XLNet adaptés au problème du dictionnaire inversé. Nos résultats sont comparés avec les résultats tirés du papier de Yan et al. provenant de modèles précédemment détaillés dans le mémoire (Yan *et al.*, 2020). L'ensemble *Définition Vue* contient des paires mot-définitions présentes dans les données d'entraînement, l'ensemble *Définition Non Vue* contient des paires mot-définitions exclues des données d'entraînement et l'ensemble *Description* contient des paires mot-définitions écrites par l'humain. Pour chaque ensemble de données tests, trois colonnes représentent respectivement le rang médian (meilleur=0), la précision@1/10/100 (meilleur=1) et la variance du rang (meilleur=1).

Outre le fait que DistilBERT soit une version distillée de BERT, la différence entre BERT(MLM) et DistilBERT(CLS) repose dans la manière de plonger les mots et la façon dont l'objectif est modélisé. BERT(MLM) n'utilise pas le jeton `[CLS]` pour plonger les mots de la séquence d'entrée, mais plutôt les états cachés du dernier encodeur afin de prédire les masques du mot cible par concaténation de sous-mots. BERT(CLS) utilise quant à lui le jeton `[CLS]` pour plonger les mots de la séquence d'entrée afin de prédire directement le mot cible. Les deux implémentations donnent des résultats similaires ce qui suggère qu'il n'y a pas d'impact majeur sur la performance en utilisant une ou l'autre des implémentations pour des tâches de classifications semblables au problème du dictionnaire inversé.

On remarque aussi que le modèle MS-LSTM donne de bons résultats sur les définitions vues, mais peine à bien généraliser sur les définitions non vues. Aussi, selon les résultats obtenus, XLNet semble mieux généraliser le problème du dictionnaire inversé, suivi de BERT(CLS), puis GPT-2. Il est intéressant de rappeler que XLNet et BERT utilisent le concept de bidirectionnalité. Il est donc possible que ce soit un facteur améliorant la précision des prévisions pour cette tâche.

5.4.1 Analyse des prévisions erronées

Cette section a pour but d'examiner le comportement des modèles lorsqu'ils prévoient des mots distincts des mots cibles. Le tableau 5.2 contient les résultats obtenues, après formation des modèles, en calculant le nombre d'erreurs (la prévision est différente du mot cible) selon quatre tests prenant en compte la liste de premier candidats retournée par les modèles.

	k-100	k-1000	k-10000	k-50000
DistilBERT(CLS)	266	**82**	55	46
DistilGPT-2	296	106	56	**31**
XLNet	**244**	**82**	**52**	46

TABLE 5.2 – Une fois les modèles entraînés, on s'attarde aux nombre d'erreurs de prévisions fait selon une liste de k premier candidats, pour des valeurs de k fixées à 100, 1000, 10 000 et 50 000. Si le mot cible d'une définition n'est pas présent dans la liste, alors on incrémente le nombre d'erreur total pour le test k.

Il est intéressant de noter que chacun des modèles réussit à retourner un nombre minimal d'erreur pour au moins un test. Quoique pour cette analyse, le test k-50000 est négligeable puisqu'on considère une liste de 50 000 mots pour un vocabulaire de taille d'environ 108 000 mots.

Nous considérons maintenant le test k-50000 seulement et faisons l'union des erreurs afin d'examiner quels mots les trois transformeurs n'ont pu prévoir. Pour ce test, nous avons noté que XLNet, DistilBERT et DistilGPT-2 partagent les mêmes erreurs, c'est-à-dire que la taille de l'intersection des ensembles d'erreurs est égal à 31 et que la différence des ensembles d'erreurs entre DistilBERT et XLNet est nulle. De plus, il s'avère que 16 fois les trois modèles ont fait la même prévision du mot cible. Le tableau 5.3 comprend quelques unes des 15 erreurs notables, où au moins un modèle à prévu un mot erroné distinct des deux autres.

Définition	mot cible	POS cible	modèle	prévision du mot	prévision du POS
ardently or excessively desirous	devouring	NN	GPT-2	fitted	NN
			XLNet	avid	NN
			BERT	avid	NN
chocolate that contains at least 32 percent cocoa butter	couverture	NN	GPT-2	chocolate	NN
			XLNet	mocha	NN
			BERT	cacao	NN
the act of enclosing something inside something else	enclosing	NN	GPT-2	envelopment	NN
			XLNet	enclosure	NN
			BERT	envelopment	NN
coming down or downward	descending	NN	GPT-2	prone	JJ
			XLNet	descendant	JJ
			BERT	downward	NN
a material used to coat cooking utensils and in industrial applications where sticking is to be avoided	polytetra fluoro ethylene	NN	GPT-2	paint	NN
			XLNet	stuffing	NN
			BERT	dressing	NN
dried naturally by the sun	sundried	VBN	GPT-2	solar	NN
			XLNet	sunburnt	NN
			BERT	sunburnt	NN
military actions designed to influence the perceptions and attitudes of individuals, groups, and foreign governments	psyop	NN	GPT-2	lifestyle	NN
			XLNet	politics	NN
			BERT	culture	NN
the branch of genetics that studies the genetically determined variations in responses to drugs in humans or laboratory organisms	pharmaco genetics	NN	GPT-2	cytogenetics	NN
			XLNet	genetics	NN
			BERT	genetics	NN
spending resources lavishly and wastefully	squander ing	NN	GPT-2	prodigal	NN
			XLNet	prodigal	NN
			BERT	prodigality	NN
a small town in a cattle, raising area of western North America	cowtown	NN	GPT-2	ranch	NN
			XLNet	ranch	NN
			BERT	pueblo	NN
a lover who necks	necker	NN	GPT-2	cranny	NN
			XLNet	rubberneck	NN
			BERT	knocker	NN
made less severe or intense	mitigated	VBN	GPT-2	deadened	VBN
			XLNet	moderate	VBN
			BERT	deadened	VBN

TABLE 5.3 – La liste de quelques erreurs commises par nos modèles lorsque le mot cible ne se retrouve pas dans la liste des 50 000 premiers candidats. Un des modèle doit avoir aussi fait une prévision distincte des deux autres. Les POS cibles du tableau comprenent les noms communs singuliers (`NN`), les verbes au participe passé (`VBN`) et les adjectifs (`JJ`).

On remarque que l'étiquetage du POS fait par **spaCy** commet quelques erreurs. Par exemple, la paire mot-définition (devouring, *ardently or excessively desirous*) est étiquetée comme étant un nom commun singulier (`NN`), alors que cela devrait être un adjectif (`JJ`) ou un verbe au participe présent (`VBG`). On peut donc dire que l'étiquetage du POS a introduite quelques erreurs dans l'ensemble de données. On remarque aussi que les modèles peuvent prévoir un mot qui n'a pas nécessairement le même POS associé. Par exemple, DistilGPT-2 prévoit le mot *fitted* avec un nom commun singulier (`NN`), alors que *fitted* devrait plutôt être étiqueté comme un verbe au participe passé (`VBN`).

5.4.2 Discussion

Dans cette section, nous discutons de la raison des performances de notre expérimentation par rapport au problème du dictionnaire inversé, dans laquelle on obtient des résultats dans le même ordre que les modèles que nous avons présentés dans le chapitre 4. On y discute aussi de la manière avec laquelle le réseau pourrait être optimisé de façon à obtenir de meilleurs résultats.

Transformeur. L'usage de transformeurs distillés a pour avantage de diminuer le nombre de paramètres du modèles et donc de diminuer le coût computationnel et ainsi accélérer l'apprentissage (Sanh *et al.*, 2019). En accélérant l'apprentissage, paramétrer le modèle et vérifier les hypothèses devient d'autant plus accommodable. Cependant, il est permis de croire que les résultats de nos expérimentations pourraient s'en être trouvés quelque peu améliorés si nous avions utilisé des modèles plus volumineux en nombre de paramètres. À titre d'exemple, le modèle distillé DistilBERT conserve 97% des performances du modèle de base BERT en comparaison avec les données de l'étalon de référence *GLUE* (Sanh *et al.*, 2019).

Aussi, à la lumière de notre expérimentation, il est pertinent d'utiliser la sortie de l'état caché du premier jeton de DistilBERT et du dernier jeton de DistilGPT-2 afin de représenter les mots dans un espace vectoriel contextuel. Toutefois, ce n'est peut-être pas d'emblée la meilleure solution puisque, pour chaque jeton, il y a des états cachés associés. En contrepartie, nos résultats suggèrent que, pour la tâche du dictionnaire inversé, cela n'a pas d'impact dans le cas du transformeur BERT.

Dropout. La couche de *dropout* à la sortie du transformeur n'est peut être pas la stratégie optimale de régularisation, puisqu'il n'y pas de consensus scientifique quant à savoir s'il est préférable d'utiliser le *dropout*, la normalisation des lots, ou les deux (Garbin *et al.*, 2020). D'autant plus qu'à notre connaissance, aucun test utilisant un taux de *dropout* fixé à 0.4 n'a été mené. Ce genre de paramètres peut être difficile à optimiser étant donné que la phase d'apprentissage dure plus de 100 heures.

Dropmax. La couche *dropmax* est une version stochastique de la couche *softmax* qui, à chaque itération, supprime des classes qui ne font pas partie des cibles en fonction des probabilités de suppression prédéfinies pour chaque instance (Lee *et al.*, 2018). Nous émettons l'hypothèse que cela peut forcer le réseau à considérer différents synonymes ou morphèmes durant la phase d'entraînement et ainsi améliorer la performance générale du réseau.

Aussi, par définition, un mot ne peut pas être inclus dans sa propre définition. En utilisant la couche *dropmax*, on peut supprimer complètement cette possibilité en assignant à 0 la probabilité qu'un des mots d'une définition soit identique au mot associé à la définition.

Tâches auxiliaires. Des expérimentations ont été menées sur les réseaux convolutifs afin de visualiser les sorties des couches convolutives et où les premières couches différencient des traits simples, puis des formes dans les couches suivantes pour éventuellement reconnaître des objets ciblés (Zeiler et Fergus, 2014). Nous émettons l'hypothèse que ce concept peut être appliqué aux réseaux de neurones dans le domaine du TAL où les premières couches d'un réseau peuvent assimiler des règles grammaticales simples, puis peu à peu construire une morphologie menant à une compréhension du POS et ainsi de suite jusqu'à une compréhension d'une sémantique développée dans les dernières couches. À la manière du modèle InceptionNet (Szegedy *et al.*, 2017) où des classifieurs auxiliaires sont introduits au milieu du réseau afin d'aider le gradient à se maintenir, il pourrait être judicieux de connecter le classifieur auxiliaire du POS à la sortie d'un des premiers encodeurs ou décodeurs.

Minimum local. Il est possible que le réseau ait atteint un minimum local par rapport à la fonction de coût. Augmenter le nombre d'époques peut mener le réseau à trouver un meilleur point minimal par rapport à la fonction de coût et donc, donner de meilleurs résultats en termes de performance.

Surapprentissage. Il est probable que certains mots se voyant octroyer de très mauvaises prévisions continuent de s'aggraver. Cela conduit à une augmentation de la perte alors que la précision reste inchangée et cela se traduit en une augmentation de la variance du rang. Une hypothèse est que le réseau commence à surapprendre des patrons pertinents pour l'ensemble d'apprentissage mais peu propices à la généralisation, puisque certains mots de l'ensemble de validation sont très mal prédits. En revanche, il continue d'apprendre certains patrons qui sont utiles à la généralisation, car de plus en plus de mots sont correctement classés.

Partie du discours. Après l'étiquetage du POS des mots cibles, 76% de ceux-ci sont étiquetés comme des noms communs singuliers (`NN`), 11% des adjectifs (`JJ`), 4% des verbes au participe passé (`VBN`), 2% des adverbes et 2% des verbes à la forme de base (`VB`). Comme le démontre l'analyse des erreurs de prévisions, cela semble conduire le modèle à prévoir des noms communs singuliers pour des mots devant être étiqueté autrement. Un ensemble mieux équilibré peut conduire à de meilleurs

résultats (Laurikkala, 2001). Toutefois, en se faisant, la comparaison de nos modèles aurait été biaisé puisque nous utilisons un ensemble de données de référence. Aussi, étant donné que **spaCy** introduit quelques erreurs d'étiquetage, il est envisageable qu'une correction manuelle de l'étiquetage des POS pourrait conduire à de meilleurs résultats.

5.5 Utilisation des modèles entraînés

Une application du problème du dictionnaire inversé est de mesurer la qualité d'une définition produite par un humain. Globalement, on pourrait qualifier une définition comme étant exacte si elle est située dans le top-1 des prévisions, satisfaisante si elle est dans le top-10, médiocre si elle est située dans le top-100 et inacceptable si on la retrouve hors du top-100. Une façon d'obtenir une valeur plus significative d'une définition consiste à considérer le degré de certitude associé à la prévision du modèle pour un mot cible et une définition donnée. Une autre façon d'évaluer la qualité d'une définition consiste à considérer le rang du mot cible prévu par le modèle.

En considérant avoir obtenu un résultat de rang et un degré de certitude par le modèle pour toutes les définitions contenues dans un dictionnaire D, cela peut se traduire par les équations :

$$\text{qualité}_{certitude}(D) = \text{moyenne}(\text{certitude}), \tag{5.1}$$

$$\text{qualité}_{rang}(D) = 1 - \frac{\text{moyenne}(\text{rangs})}{|V|}. \tag{5.2}$$

5.5.1 Évaluation des jeux du dictionnaire

Cette section apporte des précisions sur les résultats d'évaluations des dictionnaires recueillis à l'aide du *jeu du dictionnaire*. Quelques dictionnaires sont présentés en annexe.

Le tableau 5.4 contient nos résultats d'évaluation des dictionnaires en utilisant les transformeurs DistilGPT-2, DistilBERT et XLNet entraînés dans le but de résoudre le problème du dictionnaire inversé.

Modèle	Q1	Q2	Q3	Q4	Merriam's Webster
DistilBERT(CLS)	.020/20013/1576	.040/12712/382	.053/11275/208	.062/**9710**/**127**	.55/12/0
DistilGPT-2	.021/17268/2199	.035/12177/722	.047/10591/498	.056/9828/311	.58/12/0
XLNet	.022/26377/2971	.044/17427/669	.056/15070/286	**.068**/12482/133	.53/12/0
			certitude	rang moyen	rang médian

TABLE 5.4 – Un jeu de données de 237 dictionnaires produits avec le *jeu du dictionnaire*. Le plus petit contient 5 mots, alors que le plus grand contient 447 mots. La distribution du nombre de mots des jeux se retrouve à l'intérieur de trois quartiles ayant respectivement 24, 64 et 111 mots. Les modèles DistilBERT(CLS), DistilGPT-2 et XLNet ont été entraînés à partir du problème du dictionnaire inversé et évaluent la qualité des dictionnaires. La qualité d'un dictionnaire est évaluée selon les métriques du degré de certitude moyen/rang moyen/rang médian, en agrégeant les résultats obtenus par les modèles pour toutes les paires mots-définitions d'un dictionnaire. Nous agrégeons les résultats selon le quartile où se retrouve un dictionnaire, c'est-à-dire que *Q1* contient les résultats des dictionnaires ayant 23 mots et moins, *Q2* des dictionnaires entre 24 et 63 mots, *Q3* des dictionnaires entre 64 et 110 mots et *Q4* des dictionnaires ayant 111 mots et plus. Le dictionnaire *Merriam's Webster* est ajouté à notre évaluation à titre comparatif.

On remarque que DistilBERT(CLS) a tendance a attribué des meilleurs scores puisque ses previsions se classent avec de meilleurs rangs médians que les deux autres modèles. Étant donné que les dictionnaires du *jeu du dictionnaire* ont des définitions assez courtes, cela peut avoir comme effet de favoriser les modèles bidirectionnels tels que DistilBERT(CLS). On remarque aussi que le rang médian des prévisions sur le *Merriam's Webster* est de 0 et que le rang moyen est de 12, c'est-à-dire que les modèles prévoient pratiquement toujours le mot associé à la définition. Cependant, ce résultat est biaisé puisque les données de ce dictionnaire sont contenues dans les données d'apprentissage.

5.5.2 Discussion

Dans cette section, nous discutons de l'emploi de nos modèles à des fins d'évaluation de la qualité d'un dictionnaire.

Degré de certitude comme métrique de qualité. Nous estimons que le degré de certitude n'est pas une métrique adéquate afin d'évaluer la qualité d'un dictionnaire puisque pour le dictionnaire *Merriam's Webster* on obtient un degré de certitude moyen de 0.55 alors que le rang médian est de 0. Cela est principalement dû au fait que le modèle considère des synonymes dans son analyse des probabilités. Par exemple, pour une définition $P = \langle un, fruit, rouge \rangle$, le modèle prévoit des degrés de certitude similaires pour les mots *cerise* et *pomme*. Une façon de mitiger l'incertitude consiste

à employer le dictionnaire *Merriam's Webster* comme étalon de base. On peut ensuite pondérer la moyenne des degrés de certitude des autres dictionnaires par la moyenne du degré de certitude de l'étalon. Mais cette méthode est biaisée, puisque la majorité des définitions des dictionnaires anglophones se retrouvent dans les données d'apprentissage.

De plus, puisque nous choisissons le degré de certitude associé au *definienda* d'une définition, l'équation 5.1 ne tient pas compte de l'ambiguïté des définitions pouvant être associées à plusieurs mots. Une solution pouvant mitiger cela consisterait à faire une somme pondérée du degré de certitude selon k candidats d'un ensemble de synonymes, où k est un paramètre qui pourrait être fixé selon la taille moyenne des *synsets* retournés par WordNet.

Rang moyen comme métrique de qualité. Nous estimons que le rang moyen constitue une métrique davantage adaptée à la mesure de qualité d'un dictionnaire. Cependant, l'équation 5.1 se référant à la qualité du dictionnaire selon le rang peut être sujette à de hautes variances dans le cas où un bon rang pour une définition est attribué à un mot contenu dans un dictionnaire contenant peu de définitionss. Cela peut donner des résultats discutables en prenant, par exemple, un dictionnaire de 5 mots qui s'est vu donné une note qualitative évaluée à 0.97 selon l'équation 5.1 en utilisant le modèle DistilBERT(CLS).

Taille du vocabulaire. Nous avions émis l'hypothèse que la qualité d'un dictionnaire est proportionnelle au nombre de mots qu'il contient, selon le principe qu'en utilisant plusieurs mots, on a la possibilité de bâtir une sémantique lexicale plus approfondie. Or, cela ne s'avère pas nécessairement puisque nous avons observé qu'un dictionnaire situé dans le premier quartile peut avoir un meilleur rang moyen qu'un dictionnaire située dans le dernier quartile. Il ne nous est pas possible d'observer une tendance significative puisque dans notre ensemble de données concernant le *jeu du dictionnaire*, on n'en dénombre aucun contenant plus de 447 mots. Évidemment, la taille du vocabulaire influe sur la moyenne et donc, plus la taille du vocabulaire est grande, plus les résultats tirés des transformeurs sont *noyés* dans la moyenne. De ce principe, on peut supposer que les petits dictionnaires sont davantage sujets à être soit plus ou moins *bons*, soit très *mauvais*.

CONCLUSION

Nous avons exposé le problème du dictionnaire inversé qui consiste à trouver un mot à partir de sa définition et le *jeu du dictionnaire* comme étant un jeu sérieux ayant comme objectif d'illustrer le lexique mental.

La problématique entourant le jeu du dictionnaire et le problème du dictionnaire se rapporte à des tâches se concernant la linguistique computationnelle que nous avons divisée en six catégories soit : (**i**) le traitement du texte, (**ii**) l'analyse morphologique, (**iii**) l'analyse syntaxique, (**iv**) la sémantique lexicale, (**v**) la sémantique relationnelle et (**vi**) l'analyse discours.

Il est possible de résoudre le problème du dictionnaire inversé en combinant la résolution de certaines de ces tâches à l'aide de méthodes statistiques dont les arbres d'analyse syntaxique, les matrices de similarité et la méthode hongroise. Il est aussi possible de résoudre ce problème à l'aide de représentation des mots dans des vecteurs de nombres réels que l'on nomme *plongements de mots*. Les méthodes pour générer les plongements utilisent notamment des représentations explicites, des modèles probabilistes, des matrices de cooccurrence et des réseaux de neurones. Les plongements de mots obtenus à l'aide de réseau de neurones ont l'avantage d'être moins épars et prennent en compte le contexte des mots.

Les RNN, les LSTM et les transformeurs sont des réseaux des neurones plongeant les mots dans des représentations vectorielles à hautes dimensions. Les réseaux RNN et LSTM s'appuient sur des états cachés passés afin de capturer les dépendances entre les mots. Les transformeurs traitent plutôt une phrase dans son ensemble à l'aide d'un mécanisme d'auto-attention. Ce mécanisme utilise trois vecteurs, soit le vecteur Clé k, le vecteur Valeur v et le vecteur Requête q. Ces trois vecteurs sont obtenus à partir de trois projections linéaires à l'aide des vecteurs de poids w_k, w_v et w_q et des plongements de mots associés à la séquence d'entrée. L'idée générale du mécanisme d'attention est de calculer des produits scalaires entre q (le mot courant) et les k (les mots de contextes). Les produits scalaires obtenus entrent dans une fonction *softmax* (probabilité qu'un mot soit associé à un mot de contexte) et le résultat est multiplié au vecteur de valeur v de chaque mot, puis on fait leur somme. Puisque le *softmax* tend à donner davantage d'importance au mot courant par rapport à son propre contexte, on emploie le mécanisme d'attention avec des représentations multivectorielles

(multitêtes) des vecteurs k, q et v.

BERT est un modèle de langue de type transformeur bâti sur une pile d'encodeurs. Il est préentraîner sur des données textuelles non étiquetées en fusionnant le contexte gauche et droit des séquences d'entrée. BERT adapte la représentation d'un mot en prenant en compte son contexte. Aussi, chaque prévision de jeton masqué est conditionnée par le reste des jetons de la phrase, c'est-à-dire que la a sortie est générée de manière non autorégressive où les prévisions des jetons de masque se font au même moment. BERT converge mieux avec l'optimiseur AdamW puisque cet optimiseur y incorpore une régularisation des poids.

Le modèle GPT est un modèle de langue de type transformeur bâti sur une pile de décodeurs. Les décodeurs emploient une matrice de masquage qui empêche le contexte droit d'être fusionné à la représentation vectorielle. Aussi, GPT emploie une variante du décodeur où celui-ci crée lui-même les valeurs des vecteurs k et q.

Outre le fait que le transformeur GPT emploie des décodeurs, le transformeur GPT diffère aussi de BERT puisqu'il est autorégressif. Ainsi, après la prévision d'un jeton, ce même jeton est ajouté à la séquence d'entrées et cette nouvelle séquence devient l'entrée de la prévision à venir.

BERT néglige les dépendances entre les jetons masquées puisqu'il ne les prend pas en considération lors de ses prévisions. Le modèle XLNet est un modèle autorégressif qui surmonte les limitations de BERT en prenant en compte les contextes bidirectionnels des mots. Cela est fait en permutant l'ordre de factorisation des mots la séquence d'entrée à l'aide d'une attention masquée à double-flux.

Nous avons introduit un cadre qui parvient à une compréhension du langage naturel via un modèle basé sur des transformeurs afin de résoudre la tâche du problème du dictionnaire inversé. En utilisant des transformeurs préformés, nous bénéficions de leurs capacités à traiter les dépendances à longue portée et transférons leur apprentissage avec succès. Pour cette tâche, notre travail suggère qu'il est possible d'obtenir des gains de performances et offre des indications sur ce qui fonctionne lorsque cela implique des modèles de type transformeurs et des ensembles de données basés sur les dictionnaires.

Un autre objectif de ce mémoire était d'introduire une approche formelle englobant l'analyse informatique de la qualité sémantique des lexiques mentales. Nous concluons que le degré moyen de

certitude du réseau de neurones n'est pas une métrique significative permettant de qualifier un dictionnaire. La notion de rang moyen est plus significative bien que son interprétation dans une fourchette à pourcentage soit ambiguë. Nous observons que la taille du vocabulaire d'un dictionnaire n'a pas obligatoirement un impact sur la qualité de celui-ci. Nous pensons que cette observation peut être appréciable dans un contexte psychosocial puisque cela suggère qu'il est possible de communiquer sans ambiguïté tout en ayant un vocabulaire restreint.

Les travaux en cours et futurs comprennent les éléments suivants :

Collecte de données. Les définitions du dictionnaire ont une structure grammaticale très spécifique, vraisemblablement plus simple et plus limitée que le cas général du texte libre. Il serait donc envisageable de considérer des sources de données plus générales comme des dictionnaires urbains ou des dictionnaires de mots croisés.

De plus, puisque plusieurs définitions de sources anglophones, par exemple le *Merriam's Webster*, se retrouvent dans les données d'apprentissage, il serait préférable de prendre une référence qui n'est pas biaisée afin de normaliser le degré de certitude moyen d'un dictionnaire.

Tâches auxiliaires. Ajouter des tâches auxiliaires augmente la flexibilité du modèle et réduit le risque de surapprentissage. Hormis le POS, il est envisageable d'ajouter des tâches de prévision, par exemple, de la catégorie, de l'étymologie et du sémème du mot cible

Problème du dictionnaire inversé musical. La musique est un langage comprenant une sémantique complexe et la relation d'un mot définit par ses définitions est homologue à un accord définit par ses notes puisqu'on doit, entre autres, considérer le contexte de la phrase musicale, la tonalité, le style et les ornementations. La tâche du dictionnaire inversé pourrait être reflétée dans le domaine musical en une tâche de solfège, où un modèle devrait deviner un accord à partir des notes qui lui sont fournies. Même s'il s'agit de traitement de signaux sonores et non de texte, les transformeurs demeurent adéquats pour ce genre de tâche, puisque cela nécessite de capturer des dépendances de longue portée. Nous croyons qu'un modèle capable de résoudre cette tâche serait en mesure de discerner la sémantique harmonique et susceptible d'analyser des segments musicaux.

accept : the act of approving something
act : used when referring to an action
action : refers to something that is done
alive : possessing life when something is not death
amount : used as a synonym of quantity
appearing : the act of seeming like something
approving : when you accept something
attributes : something that refers to categories
base : word used as a synonym of support
behaving : to act a certain way
belonging : something that is in the appropriate place
big : refers to something that is great
born : the act of standing something
certain : synonym of the word some
characteristics : something that refers to attributes
choosing : the act of selecting something
combining : the act of putting together two or more things
complete : synonym of the word total
complicated : said of something that is difficult
composed : to make or form by combining things
controlled : to have power over something
creating : the act of forming something
culture : something that is learned and passed through generations
definitions : refers to the meaning of something
difference : when two things are not similar
difficult : said of something that is complicated
dividing : the act of separating something
else : used when referring to something different
ending : when something is finished
entirely : word used as a synonym of completely
exactly : refers to something that is precise
explanation : an statement given about why something is the way it is
final : refers tot he last part
forcing : the act of imposing something on someone
form : word used as a synonym of make
frequency : refers to the amount of times something is repeated in time
function : refers tot he role that something plays
getting : the act of receiving something
go : to move to a certain location
grant : to give something to someone
group : more than two persons
happens : verb used to say that something occurs
horse : animal that humans ride
how : used to say in what manner something is done
humans : species of animal that possess reason
information : new knowledge that is acquired
king : the ruler of a kingdom and the people in it
knowledge : information about varied things
learned : the act of acquiring knowledge
life : refers to the opposite of death
living : the act of being alive and not death
location : refers to a specific place or region
making : the act of creating something
meaning : refers to the definitions of something
measured : to give an amount to something
mechanically : used when referring to something with a mechanical structure
mess : said of a situation that is difficult or complicated
modifying : the act of changing something
more : refers to something that is in greater amount
multiple : used to talk about more than one thing
new : when something is novel
norms : a rule imposed by a society
number : a word symbol that represents a specific amount
occurs : when something happens
often : said when something occurs with frequency
opposite : refers tot he complete difference of something to something
organism : living being composed of specific organs
other : used to denote difference between two things
own : word that signifies possession
owning : the act of having something
particular : used to refer to something that is specific
people : name used when referring to a human being
physical : referring to the body of a person
placement : used to refer tot he location of something
plays : the act of performing a role
possess : the act of having something
possession : word that signifies ownership
precise : word used as synonym of exact
quantity : used to refer to the amount of something
rational : referring to something that uses reason
receiving : the act of getting something from somebody
referring : word used as synonym of allude
remain : the act of staying in a certain way
represents : the act of giving meaning to something
ride : the act of being on something for transportation
ruled : to have power over a country
rules : culturally accepted norms
same : when something is exactly similar to another thing
second : refers to something that is in position number two
section : refers to a part of something
selected : the act of choosing something
shape : refers to the form of something
signifies : the act of meaning something
situation : a particular place or location
small : refers tot he opposite of big
some : synonym word for certain
someone : used when referring to a person
sounds : refers to a noise made by something
speaking : used as a synonym of talking
specific : used to refer to a particular thing
statement : refers to an explanation given to something
stops : the act to not do something
structure : refers to something that is constructed
symbol : something that symbolizes something else
synonym : a word that mess the same thing as another word
things : different way of referring to an object
together : refers to doing something in or into a group
total : synonym of the word complete
transportation : refers to something that transports something form one location to another
two : refers to the second number
used : to do something with something to get something
usually : something that occurs frequently but not all the time
verb : word that refers to action
way : refers to how something is done
where : used to say in what location something is happening
women : feminine version of a human being
:

ability : a person s skill to do something
acquiring : the act of getting something
act : verb used to refer to something that is done
adequate : said of something that is appropriate
allude : the act of making reference to something
animal : organism belonging to one of the five living kingdoms
appropriate : what we say when something is adequate
area : a specific place or region
authority : the legitimate power to do something
based : the act of having something as a support
being : refers to a living thing
bend : word that is the opposite of straight
body : the physical structure of a human being
categories : groups separated by certain characteristics
change : the act of modifying something
child : words used as a synonym of tot
combination : the result of combining two or more things
common : refers to something that is shared
completely : word used as a synonym of entirely
complicated : the act of making something difficult
constructed : word used as a synonym of make
country : region that is controlled by its own government and usually has its own culture
culturally : when something refers to culture
death : the ending of someone s or something s life
denote : the act of giving meaning to something
different : used when referring to something that is not similar
disapprove : the act of not approving of something
do : the act of performing an action
end : the last part of something
ends : the last part of something
exact : word used as synonym of precise
explains : the act of giving a reason to something
feminine : used when referring to women
finished : when something ends or stops
form : refers to the shape of something
forming : refers to the synonym of create
frequently : when something happens often
generations : group of people born and living during the same time
giving : to grant something to someone
government : selected officials that rule a country
greater : refers to something that is big
habit : refers to a usual way of behaving
having : the act of owning something
hours : refers to a measure of time
human : something referring to people
imposed : the act of forcing something onto someone
kind : a group of beings belonging to the same type
kingdoms : country ruled by a king or queen
last : refers to the end of something
legitimate : something that seems rational based not the rules in society
living : what we say when something is alive
locating : the act of placing something
make : the manner in which something is made
manner : used as synonym to way
measure : the amount given to something
mechanical : used to refer to something with a mechanism
mechanism : refers to a mechanical organism
minutes : refers to a measure of time
more : said when something has multiple things
move : the act of mechanically transferring something to another place
name : word used to allude to something specific
noise : refers to a sound made by something
novel : when something is new
object : different way of referring to a thing
officials : someone that has a position of authority
opposed : to go against something that you disapprove of
opposite : something that is completely different to something else
organs : a part of the body that has a particular function
outcome : refers tot he final result
ownership : word that signifies possession
part : refers to a section of something
passed : the act of transferring something form someone to somebody else
performing : the act of doing something
place : a specific area or location
placing : the act of locating something
position : used to refer to the location or the placement of something
possessing : the act of owning something
power : the ability to do things without being opposed
putting : the act of locating something in a position
queen : feminine ruler of a kingdom
reason : an statement that explains why something is the way it is
reference : the source of information about something
region : used to denote a specific location
repeated : the act of doing something more than one time
result : refers to the final outcome
role : the part that a person has in a particular situation
ruler : the person who rules a country
s : used to symbolize possession
say : the act of speaking to someone
seconds : refers to a measure of time
seems : the act of appearing like something
separated : the act of dividing something
shared : to have something in common
similar : two or more things that have the same characteristics
skill : word used as a synonym of ability
society : group of people that habit a common location
somebody : used to refer to a person other than yourself
something : used when referring to a specific thing
source : the place where you get something
species : refers to a kind of being
standing : to remain in a straight position
staying : word used as as a synonym of remaining
straight : refers to a position without a bend
support : word used as a synonym of base
symbolizes : the act of meaning something
talking : the act of speaking to someone
time : thing that is measured in seconds minutes and hours
tot : word used to refer to a small child
transferring : to change the location of something from one area to another
transports : something that transfers something from one place to another place
type : a group of beings belonging to the same kind
usual : when something is done with frequency
varied : refers to something that is done in multiple ways
version : refers to a specific form of something
when : used to say at what time something is happening
why : used to explain the reason of something being done
word : a combination of sounds that have a meaning

alive : an animal or person who is living
arm : a limb on an animal or person
awareness : consciousness of the existence of something
between : something separating two things
came : to go closer to something
closer : a short distance from something
condition : a state of something or someone
consciousness : state of being aware of surroundings
control : to move something in space
create : to make something new
develop : something that is developed in time
development : something that is evolving in time
distance : a measurement in length between things
essential : a necessary object for something to take place
exactly : something that is in precise terms
express : conveying information about something
feed : to give food to a living being
final : coming to an end of something
give : hand over something to someone
grow : evolving of size or state
having : when someone possess something
information : known facts about something
knowledge : awareness of information about something
known : to have knowledge about something
legs : limbs on an animal or person
limbs : an arm or leg on an animal or person
living : an animal or person who is alive
long : something that has a measure with a big distance
matter : something made up of physical substance
measurement : the length or size of something
movement : to go from one place to another in space
new : something that just came into existence
objective : something that is sure to be in existence
part : a particular dimension of something
person : a living being with consciousness and that feeds on organic matter to be alive
place : a particular point in space
possess : when someone has something
precise : something exactly expressed
progression : a state moving in development
reality : objective existence of a state
s : a term used to describe something
separating : things with a distance between them
sit : to put one s self on something
size : measurement something s dimensions
someone : an unknown person in existence
somewhere : a particular physical space
state : a particular condition of something or someone
sure : something certain to be in existence
taking : when someone possesses something
terms : a word used to describe a thing
time : a measurement of the progression of existence
unknown : something someone is not aware of
used : to utilize something to accomplish something
when : something at a particular time
:

accomplish : develop something till its end
animal : a living being that feeds on organic matter
aware : having knowledge about a situation
being : a living thing with conscience
big : something that is of considerable size
certain : something sure to be in existence
combining : putting things together to make something
conscience : awareness of one s existence
considerable : something that is big in size
conveying : make information about something knowledgeable
describe : expressing information about something
developing : to grow and evolve in time
dimensions : measurement of size or length of something
end : the final part of something
evolving : something that is developing in time
existence : something that has an objective reality
facts : known information about something
feeds : to give food to a living being
food : substance that beings feed on to live
go : to move from one place to another in distance
hand : to give something to someone
horse : an animal with four long legs used for riding
just : exactly and precisely that thing
knowledgeable : to have knowledge about things
known : to have knowledge of something s existence
length : measurement of a distance of a thing
living : an animal or person who is alive
living : an animal or person who is alive
made : combining things to create something
measure : something used to express a size
move : to go somewhere from one place to another
necessary : something essential for something to take place
object : an physical thing in existence
organic : something made of living matter
particular : something precise in existence
physical : something objective and tangible
point : a particular place on something
practical : the use for something in particular
precisely : something exactly expressed
putting : to move something onto some place
riding : to sit on and control movement of something
self : a person s essential living being
short : something that has a measurement with a small distance
situation : a state or place someone is in
small : something that is not of considerable size
something : an unknown person in existence
space : existence taking place in dimensions
substance : a particular physical matter
surroundings : the things and conditions around something
tangible : something objective and physical
thing : something objective in existence
together : things with no distance between them
use : practical dimension of an object
utilize : practical use of something
word : a term used to describe something

describe : information to give detail
detail : specifics used to talk about something
detailed : description specific not vague
else : not the something you talked about
information : detail to describe something
something : a vague description of a thing
specifics : details provided to give a description
tells : to talk about something in detail
time : what a clock tells you about
vague : describe something not specific
:

clock : something that tells time
description : provides detail about something
detailed : describe something in specific
do : thing to use with something
give : to provide something with something else
provides : to give information about something
specific : something with a detailed description
talk : to give information about something
thing : vague description of something
used : something specific to do with something
you : something vague not detailed

action : the act of doing something
capable : a thing that can be shown by a clock
displaying : the act of showing something
has : the act of possessing a thing
quality : something being possessed and displayed
something : thing capable of displaying possessing an action or quality
time : something shown by a clock

act : to display the quality of doing an action
can : something that has the quality of being capable
clock : a thing that can show time
doing : to display the quality of act
possessing : the act of having something
show : the act of displaying something
thing : something capable of displaying possessing an action or a quality
:

exists : action of something living
living : action of valuable something
thing : valuable something that exists
:

action : exists living valuable something
living : exists valuable something
something : living action that exists
valuable : something that exists living

act : something relating to having or possessing an action
alive : being having life
being : noun describing something that is alive
describing : act related to speech to connect beings and acts
female : a human being who is not a male being
human : a being that is not an animal being a person
its : related to identity either male or female of something not human
male : an alive being who is not a female being
noun : a part of speech that is not a verb
person : a human being either male or female with an identity
possessing : the act of having something
related : the act of having something connected to its identity
speech : act related to human being identity
:

act : something relating to having or possessing an action
action : to act to possess or have and identity
animal : a being that is not a human being not a person
connected : the act of having being related
female : an alive being who is not a male being
having : the act of possessing something
identity : related to a person a human being either male or female
life : something related to animals female and male human beings
male : a human being who is not a female being
part : something connected to something related to it
possess : the act of having something
related : the act of having something connected to its identity
something : related to identity of an animal or an it
verb : a part of speech that is not a noun

BIBLIOGRAPHIE

Abney, S. et Light, M. (1999). Hiding a semantic hierarchy in a markov model. Dans *Unsupervised Learning in Natural Language Processing*.

Achananuparp, P., Hu, X. et Shen, X. (2008). The evaluation of sentence similarity measures. Dans *International Conference on data warehousing and knowledge discovery*, 305–316. Springer.

Adamopoulou, E. et Moussiades, L. (2020). An overview of chatbot technology. Dans *IFIP International Conference on Artificial Intelligence Applications and Innovations*, 373–383. Springer.

Alrehamy, H. H. et Walker, C. (2017). Semcluster : unsupervised automatic keyphrase extraction using affinity propagation. Dans *UK Workshop on Computational Intelligence*, 222–235. Springer.

Bahdanau, D., Cho, K. et Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Dans *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Baker, C. F., Fillmore, C. J. et Lowe, J. B. (1998). The berkeley framenet project. Dans *COLING 1998 Volume 1 : The 17th International Conference on Computational Linguistics*.

Balakrishnan, V. et Ethel, L.-Y. (2014). Stemming and lemmatization : A comparison of retrieval performances. *Lecture Notes on Software Engineering*, *2*, 262–267.

Baldwin, S. (2012). Compute canada : advancing computational research. Dans *Journal of Physics : Conference Series*, volume 341, p. 012001. IOP Publishing.

Bang-Jensen, J. et Gutin, G. Z. (2008). *Digraphs : theory, algorithms and applications*. Springer Science & Business Media.

Bengio, Y., Simard, P. et Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bengtson, E. et Roth, D. (2008). Understanding the value of features for coreference resolution. Dans *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 294–303.

Bentivogli, L., Clark, P., Dagan, I. et Giampiccolo, D. (2011). The seventh pascal recognizing textual entailment challenge. Dans *Text Analysis Conference (TAC)*.

Bilac, S., Watanabe, W., Hashimoto, T., Tokunaga, T. et Tanaka, H. (2004). Dictionary search based on the target word description. Dans *Proceedings of the Tenth Annual Meeting of The Association for Natural Language Processing*.

Biswal, A. (2022). Recurrent neural network (rnn) tutorial : Types, examples, lstm and more. [En ligne ; Page disponible le 20-août-2022]. Récupéré de

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

Blondin Massé, A., Chicoisne, G., Gargouri, Y., Harnad, S., Picard, O. et Marcotte, O. (2008). How is meaning grounded in dictionary definitions ? Dans *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing, TextGraphs 2008*, 17–24.

Bordes, A., Usunier, N., Chopra, S. et Weston, J. (2015). Large-scale simple question answering with memory networks. *arXiv preprint arXiv :1506.02075*.

Chen, G. et Su, J. (2021). Towards non-ambiguous reverse dictionary. Dans *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 1113–1120.

Chen, P. et Zhao, Z. (2022). A unified model for reverse dictionary and definition modelling. Dans *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, 8–13.

Chen, S. F. et Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, *13*(4), 359–394.

Chicoisne, G., Blondin-Masse, A., Picard, O. et Harnad, S. (2008). Grounding abstract word definitions in prior concrete experience. Dans *Sixth Annual Conference on the Mental Lexicon*.

Cortes, C., Mohri, M. et Rostamizadeh, A. (2009). L2 regularization for learning kernels. Dans *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 109–116.

Crawford, H. V. et Crawford, J. (1997). Reverse electronic dictionary using synonyms to expand search capabilities. US Patent 5,649,221.

Croft, W., Cruse, D. A. *et al.* (2004). *Cognitive linguistics*. Cambridge University Press.

Daniel, J. et James, H. M. (2000). *Speech and Language Processing*. Prentice-Hall.

Dasgupta, S. et Ng, V. (2007). High-performance, language-independent morphological segmentation. Dans *Human Language Technologies 2007 : The Conference of the North American Chapter of the Association for Computational Linguistics ; Proceedings of the Main Conference*, 155–163.

Datamuse (2023). Datamuse. [En ligne ; Page disponible le 20-janvier-2023]. Récupéré de https://www.datamuse.com/

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). Bert : Pre-training of deep bidirectional transformers for language understanding. Dans *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies - Proceedings of the Conference*, volume 1, 4171–4186.

Dhruv, P. et Naskar, S. (2020). Image classification using convolutional neural network (cnn) and recurrent neural network (rnn) : a review. *Machine learning and information processing*, 367–381.

Dutoit, D. et Nugues, P. (2002). A lexical database and an algorithm to find words from definitions. Dans *ECAI*, volume 45, 0–454. Citeseer.

El-Kahlout, I. D. et Oflazer, K. (2004). Use of wordnet for retrieving words from their meanings. Dans *Proceedings of the global Wordnet conference (GWC2004)*, 118–123.

Elman, I. L. (2012). Lexical knowledge without a lexicon ? *Methodological and Analytic Frontiers in Lexical Research*, *47*, 197.

Ethayarajh, K. (2019). Word analogies. [En ligne ; Page disponible le 29-août-2022]. Récupéré de `https://kawine.github.io/blog/nlp/2019/06/21/word-analogies.html`

Ethayarajh, K., Duvenaud, D. et Hirst, G. (2019). Towards understanding linear word analogies. Dans *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3253–3262.

Face, H. (2022). The ai community building the future. [En ligne ; Page disponible le 21-février-2022]. Récupéré de `https://huggingface.co/`

Floridi, L. et Chiriatti, M. (2023). Gpt-3 : Its nature, scope, limits, and consequences. *Minds and Machines*, *30*(4), 681–694.

Foundation, T. A. S. (2009). O.s. project opennlp. Récupéré de `http://opennlp.sourceforge.net`

Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, *12*(2), 23–38.

Gao, J.-B., Zhang, B.-W. et Chen, X.-H. (2015). A wordnet-based semantic similarity measurement combining edge-counting and information content theory. *Engineering Applications of Artificial Intelligence*, *39*, 80–88.

Garbin, C., Zhu, X. et Marques, O. (2020). Dropout vs. batch normalization : an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19), 12777–12815.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*(1), 1–309.

Goldberg, Y. et Levy, O. (2014). word2vec explained : deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv e-prints*, arXiv–1402.

Guite-Vinet, J., Blondin Massé, A. et Munoz, Y. (2023). Le jeu du dictionnaire. [En ligne ; Page disponible le 10-juillet-2023]. Récupéré de `https://combinat.info.uqam.ca`

Hachey, B., Radford, W., Nothman, J., Honnibal, M. et Curran, J. R. (2013). Evaluating entity linking with wikipedia. *Artificial intelligence*, *194*, 130–150.

Hanson, S. et Pratt, L. (1988). Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, *1*.

He, P., Liu, X., Gao, J. et Chen, W. Deberta : Decoding-enhanced bert with disentangled attention. Dans *International Conference on Learning Representations*.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L. et Wang, H. (2013). Learning entity representation for entity disambiguation. Dans *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 30–34.

Heidenreich, H. (2018). Introduction to word embeddings. [En ligne ; Page disponible le 20-août-2022]. Récupéré de `https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc`

Helmini, S. (2019). All you need to know about rnns. [En ligne ; Page disponible le 20-août-2022]. Récupéré de `https://towardsdatascience.com/all-you-need-to-know-about-rnns-e514f0b00c7c`

Hill, F., Cho, K., Korhonen, A. et Bengio, Y. (2016). Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, *4*, 17–30.

Hirschman, L. et Gaizauskas, R. (2001). Natural language question answering : the view from here. *natural language engineering*, *7*(4), 275–300.

Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Honnibal, M. et Montani, I. (2017). spaCy 2 :natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https ://spacy.io/.

Huang, E. H., Socher, R., Manning, C. D. et Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. Dans *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 873–882.

Huang, G., Liu, Z., Van Der Maaten, L. et Weinberger, K. Q. (2017). Densely connected convolutional networks. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Ioffe, S. et Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. Dans *International conference on machine learning*, 448–456. PMLR.

Irissappane, A. A., Yu, H., Shen, Y., Agrawal, A. et Stanton, G. (2020). Leveraging GPT-2 for classifying spam reviews with limited labeled data via adversarial training. *Computer Research Repositary (CoRR)*, *abs/2012.13400*.

Jalamar, J. (2020a). The illustred gpt2. [En ligne ; Page disponible le 21-février-2022]. Récupéré de `https://jalammar.github.io/illustrated-gpt2/`

Jalamar, J. (2020b). The illustred transformers. [En ligne ; Page disponible le 21-février-2022]. Récupéré de https://jalammar.github.io/illustrated-transformer/

Kahan, S., Pavlidis, T., Baird, H. et Mantas, J. (1987). An overview of character recognition methodologies. pattern recognition, vol. 19, no 6, p. 425-430, 1986. *IEEE T-PAMI*, *9*(2), 274–288.

Kandola, J., Cristianini, N. et Shawe-taylor, J. (2002). Learning semantic similarity. Dans S. Becker, S. Thrun, et K. Obermayer (dir.). *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Kartsaklis, D., Pilehvar, M. et Collier, N. (2020). Mapping text to knowledge graph entities using multi-sense lstms. Dans *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 1959–1970.

Kauser (2016). Assignment problem easy steps to solve - hungarian method with optimal solution. [En ligne ; Page disponible le 20-août-2022]. Récupéré de https://www.youtube.com/watch?v=rrfFTdO2Z7I

Khanna, C. (2021). Byte-pair encoding : Subword-based tokenization algorithm. [En ligne ; Page disponible le 28-juin-2022]. Récupéré de https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0

Kienzler, R. (2017). Recurrent neural networks and lstm tutorial in python and tensorflow. [En ligne ; Page disponible le 15-novembre-2021]. Récupéré de https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/

Kim, Y. (2014). Convolutional neural networks for sentence classification. Dans *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Association for Computational Linguistics.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. *et al.* (2017). Overcoming catastrophic forgetting in neural networks. volume 114, 3521–3526. National Acad Sciences.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A. et Fidler, S. (2015). Skip-thought vectors. *Advances in neural information processing systems*, *28*.

Köppen, M. (2000). The curse of dimensionality. Dans *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, 4–8.

Kozima, H. et Ito, A. (1997). The role of shared attention in human-computer conversation. Dans *ROCLING 1997 Poster Papers*, 224–228.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Kullback, L. (1951). Kullback s., leibler ra. *On information and sufficiency, The Annals of*

Mathematical Statistics, *22*(1), 79–86.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. et Soricut, R. (2019). Albert : A lite bert for self-supervised learning of language representations. Dans *International Conference on Learning Representations*.

Larousse. (2022). *Encyclopédie Larousse*. Éditions Larousse.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. Dans *Artificial Intelligence in Medicine : 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001 Cascais, Portugal, July 1–4, 2001, Proceedings 8*, 63–66. Springer.

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

LeCun, Y. (1998). The mnist database of handwritten digits. *http ://yann.lecun.com/exdb/mnist/*.

Lee, H. B., Lee, J., Kim, S., Yang, E. et Hwang, S. J. (2018). Dropmax : Adaptive variational softmax. *Advances in Neural Information Processing Systems*, *31*.

Léon, P. et Bhatt, P. (2017). *Structure du français moderne, Quatrième édition : Introduction à l'analyse linguistique*. Canadian Scholars.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone. Dans *Proceedings of the 5th Annual International Conference on Systems Documentation*, p. 24–26. Association for Computing Machinery.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. et Zettlemoyer, L. (2020). Bart : Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. Dans *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Li, B., Weng, Y., Xia, F., He, S., Sun, B. et Li, S. (2022). Lingjing at semeval-2022 task 1 : Multi-task self-supervised pre-training for multilingual reverse dictionary. Dans *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, 29–35.

Li, X., Chen, S., Hu, X. et Yang, J. (2019). Understanding the disharmony between dropout and batch normalization by variance shift. Dans *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2682–2690.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. et Stoyanov, V. (2019). Roberta : A robustly optimized BERT pretraining approach. *CoRR*, *abs/1907.11692*.

Loshchilov, I. et Hutter, F. (2018). Decoupled weight decay regularization. Dans *International Conference on Learning Representations*.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Transl. Comput. Linguistics*,

11(1-2), 22–31.

Malekzadeh, A., Gheibi, A. et Mohades, A. (2021). Predict : persian reverse dictionary. *arXiv preprint arXiv :2105.00309*.

Martin, L., Müller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D. et Sagot, B. (2019). Camembert : a tasty french language model. *CoRR*, *abs/1911.03894*.

Maulud, D. H., Zeebaree, S. R., Jacksi, K., Sadeeq, M. A. M. et Sharif, K. H. (2021). State of art for semantic analysis of natural language processing. *Qubahan Academic Journal*, *1*(2), 21–28.

Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*, volume 71. Siam.

Mikolov, T., Chen, K., Corrado, G. S. et Dean, J. (2013a). Efficient estimation of word representations in vector space. Dans *The International Conference on Learning Representations (ICLR)*.

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

Miller, G. A. (1995). Wordnet : a lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Miller, G. A., Leacock, C., Tengi, R. et Bunker, R. T. (1993). A semantic concordance. Dans *Human Language Technology : Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Moher, M. et Gulliver, T. (1998). Cross-entropy and iterative decoding. *IEEE Transactions on Information Theory*, *44*(7), 3097–3104.

Morinaga, Y. et Yamaguchi, K. (2018). Improvement of reverse dictionary by tuning word vectors and category inference. *Communications in Computer and Information Science*, *920*, 533–545.

Morise, M., Yokomori, F. et Ozawa, K. (2016). World : a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, *99*(7), 1877–1884.

Navigli, R. et Ponzetto, S. P. (2012). Babelnet : The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193*, 217–250.

Newman, B. et Aydin, A. (2022). Tip of your tongue : Methods for an effective reverse dictionary model.

Niu, Z.-Y., Ji, D. et Tan, C. L. (2007). I2r : Three systems for word sense discrimination, chinese word sense disambiguation, and english word sense disambiguation. Dans *Proceedings of*

the fourth international workshop on semantic evaluations (SemEval-2007), 177–182.

Niwattanakul, S., Singthongchai, J., Naenudorn, E. et Wanapu, S. (2013). Using of jaccard coefficient for keywords similarity. Dans *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, 380–384.

Noe (2019). Why is the decoder not a part of bert architecture ? [En ligne ; Page disponible le 20-août-2022]. Récupéré de `https://datascience.stackexchange.com/questions/65241/why-is-the-decoder-not-a-part-of-bert-architecture`

Oflazer, K., Göçmen, E. et Bozşahin, C. (1994). An outline of turkish morphology. *Report to NATO Science Division SfS III (TU-LANGUAGE), Brussels.*

Okpor, M. (2014). Machine translation approaches : issues and challenges. *International Journal of Computer Science Issues (IJCSI)*, *11*(5), 159.

OpenAi (2022). Chatgpt : Optimizing language models for dialogue. [En ligne ; Page disponible le 27-décemebre-2022]. Récupéré de `https://openai.com/blog/chatgpt/`

Palmer, D. D. (2000). Tokenisation and sentence segmentation. *Handbook of natural language processing*, 11–35.

Palmer, M., Gildea, D. et Xue, N. (2010). Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, *3*(1), 1–103.

Park, S. et Kwak, N. (2016). Analysis on the dropout effect in convolutional neural networks. Dans *Asian conference on computer vision*, 189–204. Springer.

Partridge, D. (1987). The scope and limitations of first generation expert systems. *Future Generation Computer Systems*, *3*(1), 1–10.

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

Phu, V. N., Tran, V. T. N., Chau, V. T. N., Dat, N. D. et Duy, K. L. D. (2017). A decision tree using id3 algorithm for english semantic analysis. *International Journal of Speech Technology*, *20*(3), 593–613.

Pollard, C. et Sag, I. A. (1988). *Information-based syntax and semantics : Vol. 1 : fundamentals.* Center for the Study of Language and Information.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program.*

Radford, A., Narasimhan, K., Salimans, T. et Sutskever, I. (2018). Improving language understanding with unsupervised learning. OpenAI.

Ramachandran, P., Liu, P. J. et Le, Q. (2017). Unsupervised pretraining for sequence to sequence learning. Dans *Proceedings of the 2017 Conference on Empirical Methods in Natural*

Language Processing, 383–391.

Reynar, J. C. (1998). *Topic segmentation : Algorithms and applications*. University of Pennsylvania.

Reynar, J. C. et Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. Dans *Applied Natural Language Processing Conference*.

Riegel, M. (1987). Définition directe et indirecte dans le langage ordinaire : les énoncés définitoires copulatifs. *Langue française*, (73), 29–53.

Robert, L. P. (2022). *Dictionnaire Le Petit Robert*. Editis.

Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1985). *Learning internal representations by error propagation*. Rapport technique, California Univ San Diego La Jolla Inst for Cognitive Science.

Rumshisky, A. et Stubbs, A. (2017). Machine learning for higher-level linguistic tasks. In *Handbook of linguistic annotation* 333–351. Springer.

Saeed, M. (2023). A gentle introduction to positional encoding in transformer models. [En ligne ; Page disponible le 20-janvier-2023]. Récupéré de `https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/`

Said El Hassani, A. H. (2011). Vers une représentation normalisée de la banque lexicale de l'iera. *Les Ressources Langagieres : Construction et Exploitation*.

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

Shannon, C. E. (1951). Prediction and entropy of printed english. *Bell system technical journal*, *30*(1), 50–64.

Shaw, R., Datta, A., VanderMeer, D. et Dutta, K. (2013). Building a scalable database-driven reverse dictionary. *IEEE Transactions on Knowledge and Data Engineering*, *25*(3), 528–540.

Shen, Y. et Liu, J. (2021). Comparison of text sentiment analysis based on bert and word2vec. Dans *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, 144–147. IEEE.

Siddique, B. et Beg, M. S. (2022). Adjective phrases in pnl and its application to reverse dictionary. *IEEE Access*, *10*, 28385–28396.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y. et Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Dans *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sproat, R. W. et Wilkes, M. V. (1992). *Morphology and computation*. MIT press.

Srilatha, D., Rao, D. P., Nikhil, C. et Teja, J. S. (2021). Classifying fake news articles using machine learning techniques. *Perspectives in Communication, Embedded-systems and Signal-processing-PiCES*, *5*(6), 62–67.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stevenson, M. et Wilks, Y. (2003). Word sense disambiguation. *The Oxford handbook of computational linguistics*, *249*, 249.

Strang, G., Strang, G., Strang, G. et Strang, G. (1993). *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA.

Styne, M. M. (1986). Poems by computer : Introducing poetry in a high-tech society. Education Ressources Information Center (ERIC).

Susnjak, T. (2022). Chatgpt : The end of online exam integrity ? *arXiv preprint arXiv :2212.09292*.

Szegedy, C., Ioffe, S., Vanhoucke, V. et Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. Dans *Thirty-first AAAI conference on artificial intelligence*.

Team, G. B. (2019). Tutorials word2vec. [En ligne ; Page disponible le 15-novembre-2021]. Récupéré de `https://www.tensorflow.org/tutorials/text/word2vec`

Thorat, S. et Choudhari, V. (2016). Implementing a reverse dictionary, based on word definitions, using a node-graph architecture. Dans *COLING*.

Tian, Y., Liu, P. et Du, B. (2020). Bipre : Bi-directional inter-personal relationship extraction task. Dans *Workshop on Chinese Lexical Semantics*, 640–651. Springer.

Tjong Kim Sang, E. F. et De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task : Language-independent named entity recognition. Dans *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 142–147.

Tseng, H., Jurafsky, D. et Manning, C. D. (2005). Morphological features help pos tagging of unknown words across language varieties. Dans *Proceedings of the fourth SIGHAN workshop on Chinese language processing*.

Uryupina, O., Poesio, M., Giuliano, C. et Tymoshenko, K. (2011). Disambiguation and filtering methods in using web knowledge for coreference resolution. Dans *Twenty-Fourth International FLAIRS Conference*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. et Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vauquois, B. (2003). Automatic translation. a survey of different approaches. *Readings in machine translation*, 333–337.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Wilbur, W. J. et Sirotkin, K. (1992). The automatic identification of stop words. *Journal of information science*, *18*(1), 45–55.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N. et Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, *30*.

Wold, S., Esbensen, K. et Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M. et Drachsler, H. (2021). Are we there yet ?-a systematic literature review on chatbots in education. *Frontiers in artificial intelligence*, *4*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. *et al.* (2016). Google's neural machine translation system : Bridging the gap between human and machine translation. *arXiv preprint arXiv :1609.08144*.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. et Raffel, C. (2021). mT5 : A massively multilingual pre-trained text-to-text transformer. Dans *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 483–498., Online. Association for Computational Linguistics.

Yan, H., Li, X., Qiu, X. et Deng, B. (2020). Bert for monolingual and cross-lingual reverse dictionary. Dans *Findings of the Association for Computational Linguistics : EMNLP 2020*, 4329–4338.

Yang, J. et Ma, J. (2019). Feed-forward neural network training using sparse representation. *Expert Systems with Applications*, *116*, 255–264.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. et Le, Q. V. (2019). Xlnet : Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Yin, J., Liu, Z., Jin, Z. et Yang, W. (2012). Kernel sparse representation based classification. *Neurocomputing*, *77*(1), 120–128.

Young, N. (1988). *An Introduction to Hilbert Space*. Cambridge University Press.

Zamani, H. et Croft, W. B. (2016). Embedding-based query language models. Dans *Proceedings of the 2016 ACM international conference on the theory of information retrieval*, 147–156.

Zeiler, M. D. et Fergus, R. (2014). Visualizing and understanding convolutional networks. Dans *European conference on computer vision*, 818–833. Springer.

Zhang, L., Qi, F., Liu, Z., Wang, Y., Liu, Q. et Sun, M. (2020). Multi-channel reverse dictionary model. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 312–319.

Zhang, M. (2020). A survey of syntactic-semantic parsing based on constituent and dependency structures. *Science China Technological Sciences*, *63*(10), 1898–1920.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. et Fidler, S. (2015). Aligning books and movies : Towards story-like visual explanations by watching movies and reading books. Dans *Proceedings of the IEEE international conference on computer vision*, 19–27.

Zock, M. et Bilac, S. (2004). Word lookup on the basis of associations : from an idea to a roadmap. Dans *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*, 29–35.