UNIVERSITÉ DU QUÉBEC À MONTRÉAL

EXPLORATION DES APPROCHES D'ANALYSE DES SENTIMENTS À BASE D'ASPECTS CIBLÉS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

FIRAS CHERIF

AVRIL 2022

REMERCIEMENTS

Au terme de ce travail, je tiens à remercier et à exprimer ma profonde gratitude au Professeur Aziz Salah, directeur de ce mémoire, qui a suivi mes travaux avec bienveillance et m'a prodigué sans cesse de précieux conseils. Je le remercie vivement aussi pour la confiance inébranlable à mon égard et pour ses critiques constructives qui ont certainement augmenté la qualité de ce travail.

Je voudrais exprimer mes sincères remerciements à ma mère, ma raison d'être, qui ne cesse de m'encourager et de faire naître en moi la boulimie du savoir. Je veux également remercier mon frère Fehd et ma soeur Fida pour leur présence et leur influence bénéfique. Il m'est agréable d'exprimer ma reconnaissance à mon cher père qui dort en paix et qui m'avait transmis son enthousiasme et sa passion scientifique.

Ma reconnaissance s'adresse également à tous ceux qui m'ont, d'une façon ou d'une autre, offert leurs services pour que je mène à bien ce travail.

Enfin, toute ma gratitude s'adresse à mon âme soeur, à ma femme Cyrine pour l'esprit de dévouement sans faille et aussi pour son aide précieuse tout au long de cette aventure.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

Les données massives possèdent un important potentiel scientifique, spécifiquement dans les domaines du forage de données, de l'apprentissage machine et du traitement des langues naturelles (NLP).

Ce travail de recherche concerne l'analyse automatique de données non structurées, extraites de la plateforme Yahoo afin d'automatiser un système d'extraction d'aspects et de classification du sentiment relatif à chaque aspect.

L'analyse des sentiments à base d'aspects ciblés (TABSA) est devenue populaire dans les domaines de la recherche et des affaires, car elle permet d'analyser un texte pour identifier différents aspects liés à une cible donnée et de déterminer le sentiment correspondant à chaque aspect détecté. Les résultats de la tâche TABSA sont plus précis qu'une analyse générale des sentiments, car l'analyse à base d'aspects ciblés examine à la fois l'aspect et son contexte dans le texte.

Dans ce travail, nous viserons à améliorer les performances des modèles d'analyse des sentiments à base d'aspects ciblés basés sur l'apprentissage par transfert. Tout d'abord, nous présenterons les différentes recherches effectuées sur la tâche TABSA. Plus précisément, nous définirons les concepts de base et analyserons les modèles d'apprentissage profond utilisés par différents auteurs sur le jeu de données Sentihood. Ensuite, nous explorerons les modèles basés sur les réseaux récurrents notamment les réseaux de longue mémoire à court terme (LSTM) et les réseaux d'entités récurrentes (EntNet). De plus, nous explorerons les modèles basés sur l'apprentissage par transfert tel que BERT-Pair. Enfin, nous proposerons deux nouveaux modèles d'apprentissage par transfert intitulés BERT-SentClass et DistilBERT-SentClass et nous comparerons nos résultats obtenus sur le jeu de données Sentihood avec les résultats explorés dans la littérature.

Mots clés : analyse des sentiments, apprentissage par transfert, apprentissage profond, BERT, données massives, DistilBERT, EntNet, forage de données, LSTM, NLP, Sentihood, TABSA.

INTRODUCTION

Introduction générale

Dans ce monde innovant, mouvementé et si rapide, chaque individu peut s'exprimer à sa guise sur toutes les plateformes exposant ainsi ses pensées, ses goûts et ses endroits préférés. Ceci permet aux internautes de partager leurs expériences sur les différentes plateformes de médias sociaux. Ce partage permet aussi aux utilisateurs de profiter des avis des autres et de prendre en considération leurs suggestions pour évaluer le produit ou le service afin de vivre une expérience meilleure et sans tracas. Cela économise non seulement leur temps, mais aussi leur argent, leur énergie et leurs efforts.

Certes, le partage d'opinion possède une multitude d'avantages, mais il crée parfois des difficultés afin d'obtenir les informations voulues à cause du tas d'informations publiées pêle-mêle. Par conséquent, beaucoup de temps et d'énergie sont gaspillés si nous ne sommes pas en mesure de trouver et d'évaluer les informations pertinentes que nous recherchons.

Auparavant, on préparait des questionnaires et des enquêtes pour obtenir les avis et les opinions des clients concernant un produit ou un service. Ce processus n'était pas efficace, non seulement parce qu'il prenait beaucoup de temps, mais aussi, parce que les clients ne répondaient pas toujours sérieusement aux questions. Ils pouvaient même laisser des réponses vides, ce qui produisait des informations incomplètes et parfois trompeuses. De plus, ces questionnaires étaient inaccessibles au public.

De nos jours, grâce aux réseaux sociaux et aux plateformes d'avis en ligne, il devient plus facile aux consommateurs de partager leurs avis et aux entreprises de les collecter et de les analyser. Dans ce contexte, l'analyse des sentiments joue un rôle essentiel. Une analyse des sentiments, également appelée extraction d'opinions, est un processus qui extrait la polarité d'une phrase (par exemple positive, négative, neutre) en appliquant des techniques d'exploration de texte et de traitement automatique du langage naturel (NLP).

L'analyse traditionnelle permet de trouver le sentiment général exprimé au niveau de la phrase ou au niveau du texte. Cependant, une phrase ou un texte peut contenir plusieurs cibles avec

différentes polarités, ce qui rend l'analyse générale des sentiments peu précise. Par conséquent, nous avons exploré la tâche d'analyse des sentiments à base d'aspects (ABSA), qui vise à déduire la polarité du sentiment relatif à un aspect spécifique.

La tâche ABSA (Wagner *et al.*, 2014; Kiritchenko *et al.*, 2014) identifie une polarité pour chaque aspect de la phrase ou du texte en entrée. Le principal défi de la tâche ABSA est de trouver la relation entre un aspect et son contexte dans la phrase avant de déduire le sentiment. Cette tâche permet d'acquérir une compréhension plus granulaire en identifiant les aspects présents dans la phrase.

Cependant, une phrase peut contenir différentes cibles. Par exemple, dans la phrase : "*La région de Montréal est plus chère, mais le choix de restaurants est plus varié que Laval.*", nous passons en revue les villes $Montréal$ et $Laval$ qui sont deux cibles distinctes. Nous constatons que le sentiment dépend à la fois des cibles $\{Montréal$ et $Laval\}$ et des aspects $\{prix$ et $restauration\}$. Le sentiment à l'égard des $prix$ à $Montréal$ est négatif mais positif en matière de choix de $restaurants$, tandis que le sentiment à l'égard des deux aspects pour $Laval$ est inversé. Par conséquent, nous avons exploré la tâche d'analyse des sentiments à base d'aspect ciblé (TABSA), qui permet d'identifier la polarité d'un aspect associé à une cible donnée. De nombreux travaux (Saeidi *et al.*, 2016; Ma *et al.*, 2018; Liu *et al.*, 2018; Sun *et al.*, 2019a) ont été proposés pour la tâche TABSA en la divisant en deux sous-tâches :

- Déterminer les aspects associés à chaque cible présente dans la phrase.
- Résoudre la polarité de chaque aspect associé à une cible.

Objectifs et contributions

Dans ce mémoire, nous nous sommes intéressés aux modèles d'analyse des sentiments à base d'aspects ciblés sur le jeu de données Sentihood (Saeidi *et al.*, 2016). L'objectif principal de notre travail est de proposer un modèle qui résout la tâche TABSA basée sur l'apprentissage par transfert. Dans ce cadre, la contribution de notre travail peut être résumée par les éléments suivants :

- Nous proposons une nouvelle méthode qui traite la tâche TABSA en utilisant BERT. Notre approche consiste à étudier différentes possibilités pour alimenter BERT.
- Nous explorons la distillation des connaissances et nous comparons les performances de BERT et de sa version distillée.

Structure du mémoire

Dans le premier chapitre, nous définirons les concepts reliés à notre sujet et les mesures de performance que nous utiliserons durant l'analyse. Le deuxième chapitre sera consacré à la revue de la littérature et aura comme but la description du jeu de données Sentihood, tout en présentant une vue globale de l'état actuel des recherches. Le troisième chapitre présentera nos expérimentations réalisées dans le but d'améliorer les performances des modèles existants. Pour finir, une synthèse des principaux résultats liés aux objectifs de départ sera présentée dans le dernier et quatrième chapitre.

CHAPITRE I

CONCEPTS PRÉLIMINAIRES

Dans ce chapitre, nous présenterons les concepts et les techniques en apprentissage profond indispensables à notre étude. D'abord, nous présenterons les différents réseaux de neurones artificiels. Par la suite, nous présenterons les plongements des mots, le mécanisme d'attention et les Transformers. Pour finir, nous présenterons les mesures de performance utilisées dans ce mémoire pour évaluer les modèles tels que la précision globale (accuracy), la F-Mesure et la surface sous la courbe (AUC).

1.1 Les réseaux de neurones

L'apprentissage profond (deep learning) est un sous-domaine de l'apprentissage automatique dans lequel la machine est capable d'apprendre par elle-même grâce à un processus d'optimisation. Inspirés de la structure du cerveau humain, les algorithmes d'apprentissage profond tentent d'analyser des données et tirer des conclusions. Pour y parvenir, l'apprentissage profond utilise des algorithmes appelés réseaux de neurones artificiels.

1.1.1 Introduction aux réseaux de neurones

Un réseau de neurones artificiel est un système inspiré du fonctionnement des neurones biologiques. Il se caractérise par son aptitude d'apprentissage par expérience d'une part, et par sa capacité de classification et de généralisation d'une autre part. En fait, le réseau de neurones fonctionne avec des nombres réels et des fonctions numériques. Nous devons donc convertir les données en représentations numériques pour les indroduire dans le réseau de neurones.

Les réseaux de neurones sont généralement organisés en couches. La première couche est la couche d'entrée ; elle est chargée de transmettre au réseau les informations à analyser. La dernière couche est la couche de sortie ; elle fournit le résultat final qui est la réponse du réseau. Les couches intermédiaires sont les couches cachées dont le nombre est un hyperparamètre à déterminer en fonction du type de problème à résoudre. La figure 1.1 ci-dessous montre un exemple de réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie.

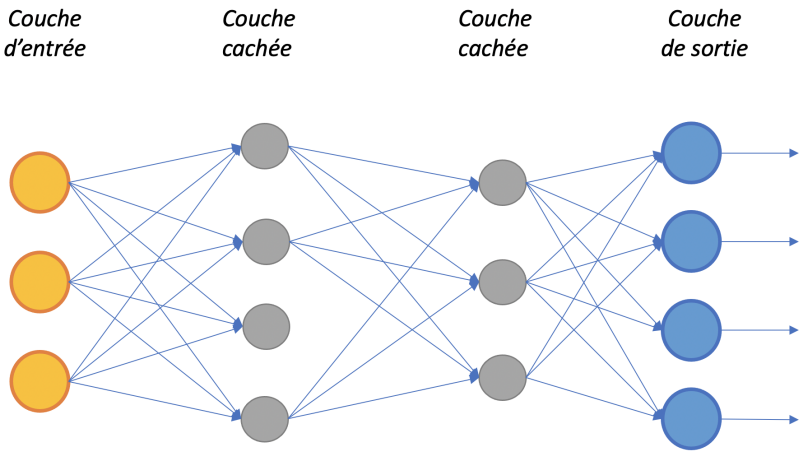

FIGURE 1.1 – Exemple d'un réseau de neurones multicouches.

Les réseaux sans couche cachée sont les réseaux les plus simples. Ils sont utiles pour les problèmes de classification ou d'approximation. Ils ont un avantage majeur, leur apprentissage converge vers une solution optimale. Ceci est dû au fait qu'il s'agit d'un système linéaire. Par contre, ils ne peuvent pas résoudre un problème non linéairement séparable (Veelenturf, 1995).

Les réseaux multicouches, également appelés perceptron multicouches, sont des modèles inventés pour résoudre des problèmes non linéaires. En général, les neurones du perceptron multicouches sont animés par une fonction d'activation non linéaire. Cette fonction d'activation décide si le neurone est activé ou non en calculant la somme pondérée de ses entrées et en ajoutant le biais.

Les réseaux multicouches classiques ont connu beaucoup de succès dans plusieurs tâches simples d'apprentissage automatique. Cependant, ils souffrent de certaines limitations liées à leur architecture. Étant donné que le nombre d'entrées d'un perceptron multicouches est fixe, il ne peut pas être entraîné sur des exemples d'entraînement de tailles variables, comme les séquences. Pour traiter des problèmes de séquences, les réseaux de neurones récurrents ont été proposés.

1.1.2 Les réseaux de neurones récurrents

Un réseau neuronal récurrent ou un RNN est un type de réseau de neurones artificiel qui possède des connexions récurrentes permettant de garder des informations en mémoire. Grâce à des boucles de rétroaction et au partage des paramètres entre les différents pas de traitement de l'entrée, le RNN permet de traiter des séquences de longueurs arbitraires et cela le rend plus adéquat pour traiter des séquences de mots. Par exemple, lors du traitement d'une séquence de trois mots, le réseau sera déroulé en trois couches cachées, soit une couche cachée pour chaque mot. La figure 1.2 représente cette idée :

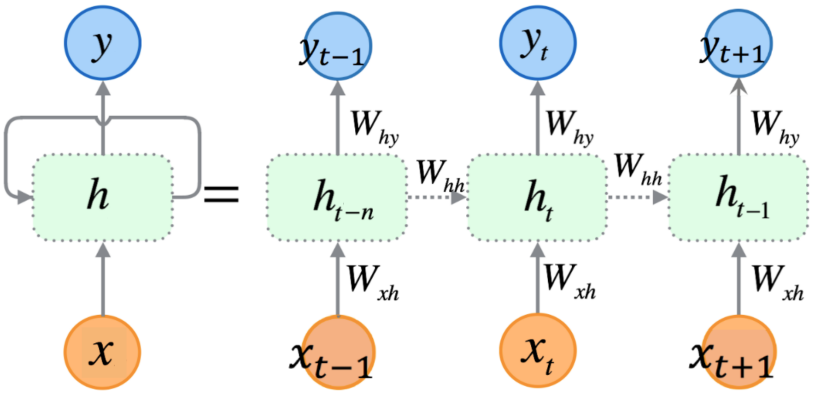

FIGURE 1.2 – Le réseau de neurones récurrent.

Le RNN utilise l'état caché h_t pour capturer l'information nécessaire des étapes précédentes. Le réseau de neurones récurrent calcule l'état caché h_t et la sortie de la séquence y_t pour un instant t en utilisant les équations suivantes :

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1.1}$$

$$y_t = g(W_{yh}h_t + b_y) \tag{1.2}$$

Dans les équations 1.1 et 1.2, W_{hx}, W_{hh}, W_{yh} désignent les matrices de poids. f et g désignent respectivement les fonctions d'activations pour l'état caché et la sortie de la séquence. b_h et b_y désignent les biais.

L'apprentissage d'un RNN est similaire à celui d'un réseau de neurones classique. En outre, la rétropropagation permet de calculer le gradient de la fonction de perte. Le gradient à chaque sortie dépend non seulement de l'étape temporelle actuelle, mais également des étapes temporelles précédentes. Si les gradients sont inférieurs à 1, alors en se multipliant consécutivement, ils ont tendance à devenir très petits et à disparaître ; c'est le problème de la disparition de gradient (Vanishing gradient). Hochreiter et Schmidhuber (1997) proposent les réseaux LSTM (Long Short Term Memory) pour résoudre ce problème et traiter de longues séquences.

1.1.3 Les réseaux LSTM

Les LSTM sont des variantes des RNN spécifiquement conçues pour traiter la problématique de la disparition du gradient (Hochreiter et Schmidhuber, 1997). Un réseau LSTM contient une cellule de mémoire qui peut stocker des informations pendant une longue période de temps. Cela rend un LSTM capable de capturer les dépendances à longue portée dans une phrase.

Le réseau LSTM introduit deux nouveaux éléments par rapport à un RNN. Premièrement, au lieu d'avoir une simple couche cachée complètement connectée, il en possède quatre couches qui interagissent les unes avec les autres au sein d'une cellule. Deuxièmement, le LSTM dispose d'un mécanisme de mémoire supplémentaire, appelé état de la cellule c. La figure 1.3 ci-dessous montre une unité d'un réseau LSTM pour un pas temporel t :

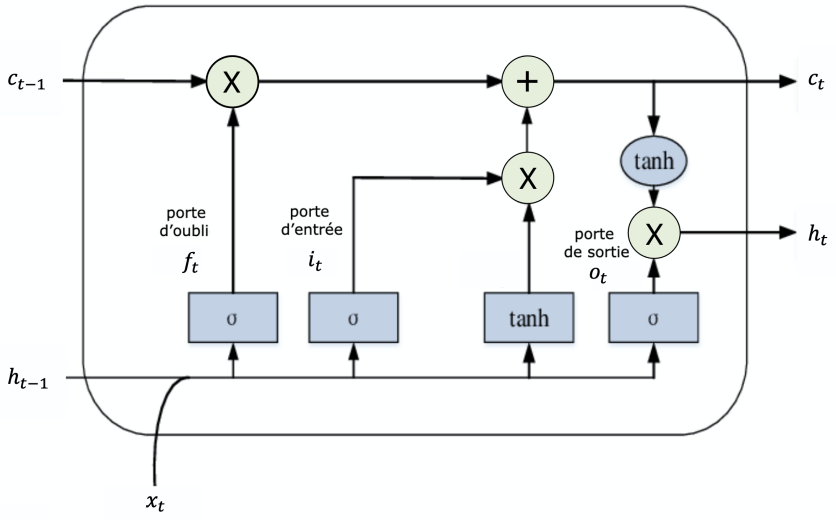

FIGURE 1.3 – Une unité LSTM pour un pas temporel t

Le LSTM se compose de trois portes qui contrôlent le flux d'informations entrant et sortant de la cellule mémoire, à savoir une porte d'entrée i, une porte de sortie o et une porte d'oubli f. Ces portes fournissent un mécanisme pour laisser passer ou effacer des informations dans la mémoire. Ainsi, le LSTM régule le flux d'informations à travers ses portes.

L'état de la cellule est représenté par c_t et la sortie de la cellule est donnée par h_t, tandis que l'entrée de la cellule est notée x_t. Le module LSTM est défini avec les équations suivantes (Mohan et Gaitonde, 2018) :

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \hat{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t * c_{t-1} + i_t * \hat{c}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(c_t) \end{aligned} \tag{1.3}$$

W_f, W_i, W_c et W_o sont les matrices de poids pour chacune des portes, σ et tanh sont respectivement la fonction sigmoïde et la tangente hyperbolique. $\hat{c}$ est l'état de cellule après sa mise à jour. Ces états sont propagés à travers le réseau et les poids sont mis à jour par rétropropagation dans le temps. La porte d'oubli joue un rôle crucial dans la réduction du surajustement (overfitting) en filtrant les informations des étapes temporelles précédentes. Cet agencement de portes et de contrôle sélectif des informations est également la principale raison pour laquelle les LSTM se distinguent par leur capacité à maintenir un flux de gradient et ne souffrent pas de la disparition de gradient (Mohan et Gaitonde, 2018) comme le cas des RNN simples.

1.1.4 Les réseaux LSTM bidirectionnels

Habituellement, un LSTM unidirectionnel ne peut utiliser que les informations contextuelles des étapes précédentes. Cependant, les LSTM bidirectionnels peuvent générer des représentations plus riches en incorporant le contexte précédent et futur à chaque pas temporel. La figure 1.4 montre l'architecture déroulée d'un LTSM bidirectionnel pour trois étapes consécutives.

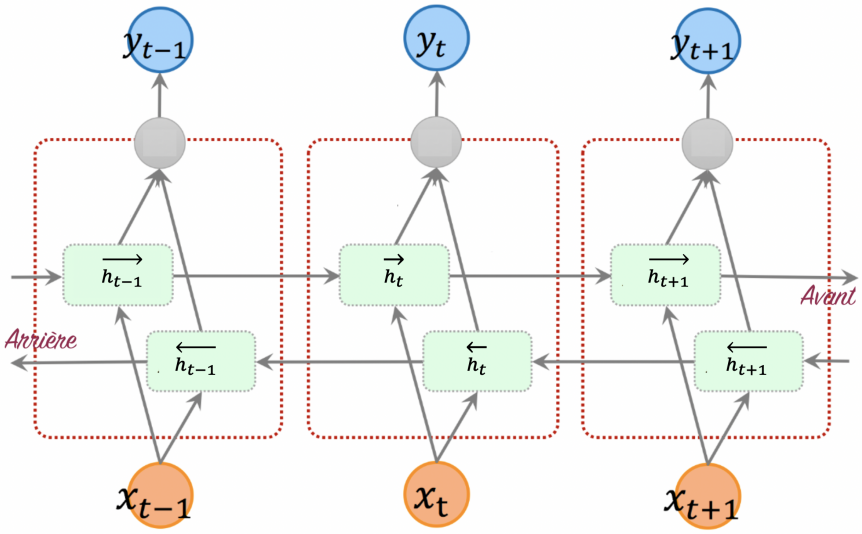

FIGURE 1.4 – L'architecture du LSTM bidirectionnel.

Les LSTM bidirectionnels (aussi appelés biLSTM) traitent une séquence de mots dans les deux directions avant et arrière, et génèrent par la suite deux vecteurs cachés. D'ailleurs, le vecteur LSTM arrière traite la séquence de mots dans l'ordre inverse et le concatène avec le vecteur

LSTM avant pour créer un état caché à chaque étape temporelle h_t (Bahdanau *et al.*, 2016) :

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \tag{1.4}$$

Où : $\overrightarrow{h_t}$ et $\overleftarrow{h_t}$ sont les vecteurs LSTM avant et arrière pour une étape temporelle t.

1.2 Les plongements de mots

Les plongements de mots sont souvent utilisés dans le traitement du langage naturel pour représenter les mots d'une phrase par un vecteur. Habituellement, les systèmes de traitement du langage naturel représentent les mots sous forme de vecteurs qui sont traités comme des symboles autonomes discrets. Ces vecteurs ne fournissent ni information contextuelle ni relations entre les différents mots de la phrase. Le modèle ne comprend pas la relation contextuelle entre les deux mots «green» et «blue» avec un encodage de type one-hot. La figure 1.5 est une illustration des vecteurs de type one-hot des deux mots «green» et «blue». La représentation d'un mot par un vecteur unique entraîne donc une rareté des données. Cela nécessite beaucoup plus de données pour réussir l'entraînement d'un modèle statistique (Larochelle *et al.*, 2008).

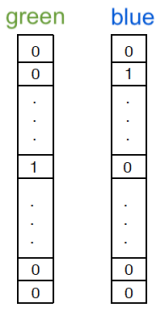

FIGURE 1.5 – Exemple des vecteurs de type one-hot des mots «green» et «blue».

Afin de surmonter cette problématique, nous pouvons utiliser des représentations vectorielles denses. D'ailleurs, en se basant sur une hypothèse distributionnelle (Kris et Ann, 2016), les modèles d'espace vectoriel peuvent être utilisés pour indiquer la similarité sémantique des mots qui ont des sens voisins. Les représentations vectorielles des mots similaires sont plus proches comme le cas de «green» et «blue», tandis que les représentations des mots très différentes sont loin dans l'espace vectoriel comme les mots «dog» et «blue» (figure 1.6).

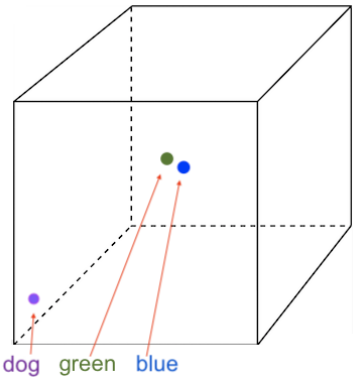

FIGURE 1.6 – Les plongements de mots dans un espace à trois dimensions.

Les approches basées sur une représentation vectorielle dense peuvent être divisées en deux catégories : les approches basées sur le comptage et les approches prédictives basées sur les réseaux de neurones (Riedl et Biemann, 2017).

Les approches basées sur le comptage utilisent les documents comme contexte et capturent la similarité sémantique entre les documents. Elles permettent de créer un vecteur de comptage des co-occurrences de mots dans un grand corpus de texte. Ces vecteurs sont ensuite projetés dans des vecteurs denses de dimension inférieure. Il existe différentes méthodes de comptage telles que GloVe (Pennington *et al.*, 2014) et TF-IDF (la mesure TF-IDF sera détaillée dans la section 3.3).

GloVe est un ensemble de vecteurs préentraînés sur un corpus de texte Wikipédia qui se base sur la matrice de co-occurrence. Chaque élément de la matrice représente le nombre de fois où le mot d'une colonne apparaît à proximité du mot d'une ligne de la matrice.

Les approches basées sur les réseaux de neurones, quant à elles, utilisent les mots voisins comme contexte pour détecter la similarité des mots. Ces méthodes prédictives ont tendance à avoir des dimensions inférieures conduisant à une meilleure représentation dense des mots. Il existe différentes méthodes qui permettent de créer ces représentations telles que Word2vec (Mikolov *et al.*, 2013b).

Word2vec est un modèle prédictif qui permet de déduire des représentations vectorielles denses de mots à partir d'un grand corpus non étiqueté. C'est un réseau de neurones qui apprend les représentations de mots en optimisant la fonction objective impliquant à la fois un mot cible et

un mot du contexte. Il existe principalement deux variantes de Word2vec, à savoir les skip-grams et les sacs à mots continus (CBOW). Les skip-grams estiment la probabilité d'apparition des mots du dictionnaire dans la même phrase pour un mot donné. Les sacs à mots continus estiment les mots du dictionnaire les plus probables pour compléter un ensemble de mots donné.

1.3 Le mécanisme d'attention

Le système de traitement visuel humain se concentre sélectivement sur certaines parties de l'image, tout en ignorant les informations non pertinentes afin d'améliorer la perception (Xu *et al.*, 2015). Puisque certains mots d'une séquence d'entrée peuvent être plus importants que d'autres, le même mécanisme peut être appliqué pour les problèmes de traitement du langage naturel. Ainsi, le mécanisme d'attention permet de se concentrer sur les mots importants de la séquence d'entrée tout en ignorant les informations non pertinentes.

Le mécanisme d'attention est apparu pour la première fois dans le domaine de la traduction de langages où la majorité des modèles utilisent l'architecture LSTM encodeur-décodeur (Vaswani *et al.*, 2017). Cette architecture se compose d'un encodeur qui encode la séquence d'entrée en une représentation que le décodeur utilise pour prédire une séquence de sortie. Dans le cas de la traduction automatique, la phrase d'entrée et le dernier mot généré dans la phrase de sortie sont pris en compte lors de la génération du mot suivant jusqu'à l'obtention du jeton indiquant la fin de la séquence.

Théoriquement, le dernier état caché d'un LSTM représente le sens de toute la phrase grâce à son mécanisme de mémorisation. Quant à la pratique, plus la phrase est longue, moins le dernier état caché du LSTM est précis pour représenter toute la phrase. Dans le domaine de traduction de texte, il est plus optimal de se concentrer en alternance sur différents mots de la phrase d'entrée pour que chaque état caché représente le mieux le dernier mot considéré par le LSTM.

Le but d'un module d'attention est de choisir, étant donné le contexte, les entrées de la phrase à utiliser pour effectuer la prochaine prédiction. En d'autres termes, un mécanisme d'attention permet au modèle de se focaliser sur les entrées importantes en leur attribuant des poids plus fort que les autres (Bahdanau *et al.*, 2016).

1.4 Les Transformers

Les Transformers ont une architecture permettant d'implémenter un modèle séquence à séquence sans impliquer de réseau récurrent (GRU, LSTM, etc.). Les réseaux récurrents étaient, jusqu'à présent, l'un des meilleurs moyens de capturer les dépendances dans les séquences. Cependant, Vaswani *et al.* (2017) ont prouvé qu'une architecture avec des mécanismes d'attention seulement (sans les réseaux de neurones récurrents) peut améliorer les résultats de plusieurs tâches en NLP.

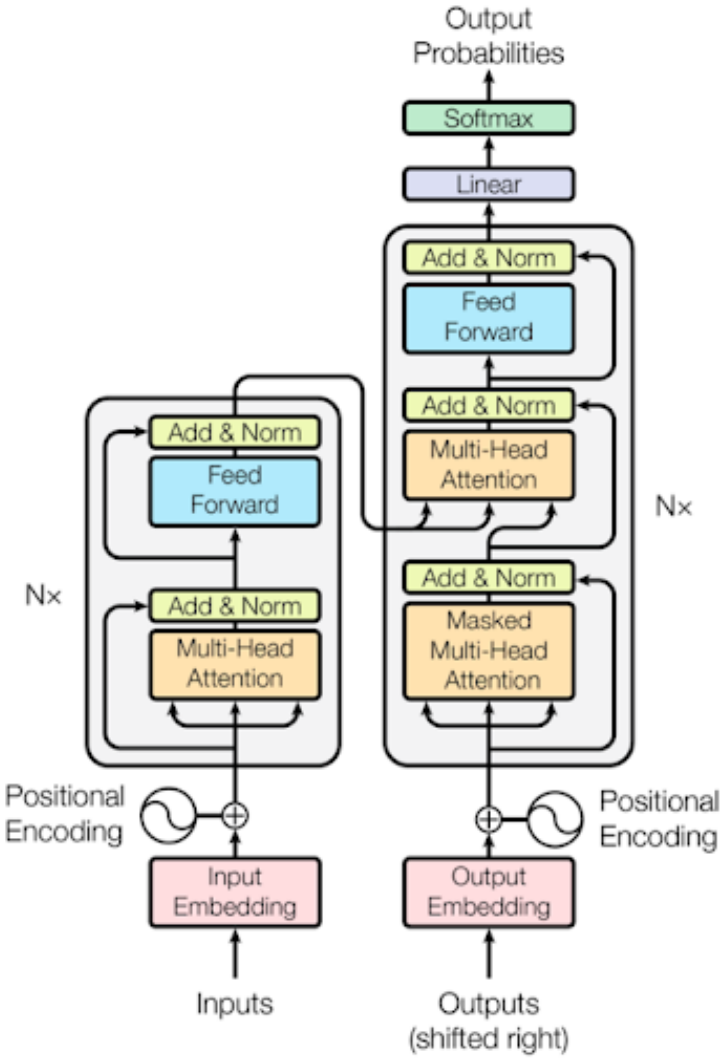

FIGURE 1.7 – L'architecture du modèle Transformer (Vaswani *et al.*, 2017).

La figure 1.7 présente l'architecture du Transformer composée d'un encodeur à gauche et d'un décodeur à droite. L'encodeur et le décodeur se composent de modules empilés les uns sur les autres plusieurs fois (Vaswani *et al.*, 2017). Un module comprend principalement des couches *Multi-Head* et *Feed Forward*. Les entrées et les sorties sont d'abord intégrées dans un espace qui représente des plongements de mots. Puisque nous n'avons pas de réseaux récurrents qui se rappelent de l'ordre de la séquence, le modèle utilise un encodage positionnel qui s'ajoute à la représentation finale.

Les Transformers utilisent le concept de *Multi-Head attention* pour suivre les informations provenant de différentes positions de la séquence. Le module d'*attention Multi-Head* calcule plusieurs sommes pondérées par le mécanisme d'attention. Chacune de ces *Multi-Head* est une transformation linéaire de la représentation d'entrée. Les différentes parties des représentations d'entrée peuvent interagir avec différentes parties d'une autre représentation à laquelle elle est comparée dans l'espace vectoriel. Cela permet au modèle de capturer différents aspects de l'entrée et d'améliorer sa capacité de généralisation. Essentiellement, l'*attention Multi-Head* n'est que plusieurs couches d'attention qui s'appliquent en parallèle, avec différentes transformations linéaires de la même entrée.

1.5 Évaluation de la performance

Pour vérifier la fiabilité et la qualité d'un modèle, il est important d'évaluer sa capacité à classer de nouvelles données qu'il n'a encore jamais vues. Il est ainsi possible de comparer différents modèles et de sélectionner le plus performant. Cette comparaison se fait en se basant sur différentes mesures telles que la précision, le rappel, la précision globale (accuracy), la F-mesure et l'AUC.

Nous distinguons deux types de classification : une binaire et une à classes multiples. Pour la classification binaire, l'entrée doit être classée dans une, et une seule, de deux classes non chevauchantes, souvent appelées classe positive 1 et négative 0. Pour la classification à classes multiples, l'entrée doit être classée dans une, et une seule, de l classes non chevauchantes.

Dans le cas de la classification binaire, la performance d'un classifieur peut être mesurée en calculant le nombre d'échantillons de classe correctement prédite (vrais positifs : TP), le nombre d'échantillons correctement prédit qui n'appartiennent pas à la classe (vrais négatifs : TN), et des échantillons qui ont été attribués de manière incorrecte à la classe (faux positifs : FP) ou qui

n'ont pas été prédits comme échantillons de classe (faux négatifs : FN) (Sokolova et Lapalme, 2009). Ces quatre catégories (TP, TN, FP et FN) sont utilisées pour calculer les mesures de performances données dans le tableau 1.1 :

Mesure	**Formule**	**Description**
Précision	$\frac{TP}{TP+FP}$	Reflète la capacité du modèle à ne pas prédire un échantillon positif comme négatif.
Rappel	$\frac{TP}{TP+FN}$	Reflète le pourcentage des échantillons positives correctement prédites.
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Reflète le taux de prédictions qui correspondent exactement aux classes réelles.
F-Mesure	$\frac{2TP}{2TP+FN+FP}$	Elle est définie comme la moyenne harmonique de la précision et du rappel.

TABLEAU 1.1 – Les mesures de performance pour la classification binaire.

Dans le cas de la classification à classes multiples, nous cherchons à attribuer un score global au lieu d'un score par classe. Pour calculer le score global, nous pouvons utiliser deux méthodes : la micro-moyenne et la macro-moyenne.

La micro-moyenne calcul le score moyen à partir des contributions de toutes les classes et ce, en accordant le même degré d'importance à tous les échantillons. En revanche, la macro-moyenne attribue les scores pour chaque classe séparément puis calcule la moyenne tout en accordant le même degré d'importance à toutes les classes. La macro-moyenne traite toutes les classes de la même manière, tandis que la micro-moyenne favorise des classes par rapport à d'autres (Sokolova et Lapalme, 2009).

Les mesures de performance micro-moyennées et macro-moyennées sont calculées en utilisant les formules présentées dans le tableau 1.2 :

Mesure	**Formule**	**Description**
$Accuracy_{moyenne}$	$\frac{\sum_{i=1}^{l} \frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{l}$	La moyenne d'accuracy est calculée à partir des accuracy de chaque classe.
$Précision_{micro}$	$\frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i+FP_i)}$	Agrégation des contributions de toutes les classes pour calculer la Précision moyenne.
$Rappel_{micro}$	$\frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i+FN_i)}$	Agrégation des contributions de toutes les classes pour calculer le Rappel moyen.
$F-Mesure_{micro}$	$\frac{précision_{micro} rappel_{micro}}{précision_{micro}+rappel_{micro}}$	Elle est définie comme la moyenne harmonique de la $Précision_{micro}$ et du $Rappel_{micro}$.
$Précision_{macro}$	$\frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FP_i}}{l}$	La moyenne des Précisions calculer pour chaque classe séparément.
$Rappel_{macro}$	$\frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FN_i}}{l}$	La moyenne des Rappels calculer pour chaque classe séparément.
$F-Mesure_{macro}$	$\frac{précision_{macro} rappel_{macro}}{précision_{macro}+rappel_{macro}}$	Elle est définie comme la moyenne harmonique de la $Précision_{macro}$ et du $Rappel_{macro}$.

TABLEAU 1.2 – Les mesures de performance pour la classification à classes multiples.

l est le nombre de classes.

En plus de l'accuracy et la F-mesure, nous allons utiliser dans ce travail une autre mesure de performance appelée AUC (en anglais ; Area Under the Curve). Elle est mesurée par la surface sous la courbe ROC (Receiver Operating Characteristic) (Fogarty *et al.*, 2005). L'aire sous la courbe ROC est une mesure qui reflète la capacité du modèle à éviter les fausses classifications. Cette mesure a été généralisée pour la classification à classe multiple par Hand et Till (2001).

1.6 Conclusion

Dans ce premier chapitre, nous avons introduit les différents réseaux de neurones utilisés dans les futures expérimentations. Nous avons également expliqué le concept de plongement de mots, le mécanisme d'attention et les Transformers. À la fin de cette partie, nous avons présenté les mesures qui seront utilisées pour évaluer la performance des modèles.

Dans le deuxième chapitre, nous présentons le jeu de données Sentihood et son processus de construction. Un état de l'art de l'application des méthodes d'apprentissage profond sur la tâche TABSA est également détaillé.

CHAPITRE II

REVUE DE LA LITTÉRATURE

Ce chapitre présente les travaux existants qui abordent la problématique de l'analyse des sentiments à base d'aspects ciblés. Tout d'abord, nous définirons les différents niveaux d'analyse des sentiments. Nous commencerons la présentation du niveau le plus général au niveau le plus détaillé. Ensuite, nous présenterons le jeu de données Sentihood et son processus de construction. Par la suite, nous présenterons les différents travaux basés sur les réseaux de neurones et l'apprentissage par transfert pour le traitement de l'analyse des sentiments ciblés. Enfin, nous clôturons ce chapitre par une analyse des différents résultats obtenus.

2.1 Les niveaux d'analyse des sentiments

L'analyse des sentiments est un domaine de recherche extrêmement actif dans le traitement du langage naturel (Kolkur *et al.*, 2015). En général, l'analyse des sentiments peut être effectuée à quatre niveaux, comme le montre la figure 2.1 :

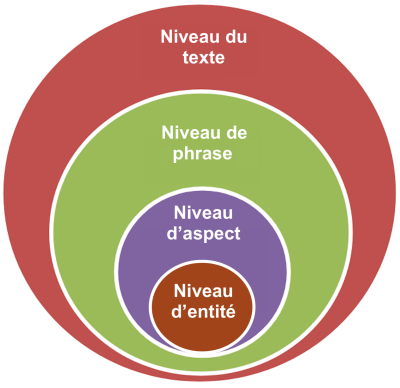

FIGURE 2.1 – Les niveaux d'analyse des sentiments.

- Analyse au niveau du texte : Aussi appelé analyse au niveau du document, la tâche à ce niveau est de déterminer le sentiment global du texte. Le sentiment peut être *positif*, *négatif* ou *neutre*. Pour ce niveau d'analyse, nous supposons que chaque texte exprime un sentiment.

- Analyse au niveau de phrase : La tâche à ce niveau est de déterminer si chaque phrase exprime un sentiment. Les phrases d'un texte peuvent être objectives ou subjectives. Pour ce niveau d'analyse, nous supposons que chaque phrase désigne un seul sentiment.

- Analyse au niveau d'aspect : La tâche à ce niveau consiste à effectuer une analyse des sentiments plus détaillée que celle effectuée au niveau du texte ou de la phrase. Le traitement commence par l'identification des aspects, suivie par l'évaluation du sentiment correspondant.

- Analyse au niveau d'entité : La tâche à ce niveau est d'identifier les aspects et les entités, suivi par l'évaluation du sentiment exprimé pour chaque pair (*entité*, *aspect*). Par exemple, la phrase *"I really liked my visit to London, but the roads are so confusing and the food is tasteless."*. Pour l'entité *London*, nous pouvons évaluer le sentiment des trois aspects : *général* (*positive*), *la gestion des routes* (*negative*) et *la nourriture* (*negative*).

2.2 Le jeu de données Sentihood

SentiHood est un extrait des réponses aux questions de la plateforme Yahoo. Ce jeu de données a été conçu par Saeidi *et al.* (2016) pour traiter la tâche d'analyse de sentiment à base d'aspects ciblés. Les réponses choisies sont relatives aux questions sur les quartiers de la ville de Londres, ce qui fait que les entités cibles de ce jeu de données sont des emplacements.

Les phrases de ce jeu de données comportent au maximum deux mentions d'emplacements et ils sont divisées en deux groupes :

- Un groupe de phrases contenant une seule mention d'emplacement «*Single*».
- Un groupe de phrases contenant deux mentions d'emplacement «*Multi*».

Le jeu de données Sentihood contient au total 5215 phrases, dont 3862 phrases qui contiennent un seul emplacement et 1353 phrases qui contiennent deux emplacements. Chaque phrase peut contenir une ou plusieurs opinions. La figure 2.2 montre deux phrases du jeu de données qui contiennent respectivement une seule opinion et deux opinions.

```
{
    "opinions": [
        {
            "sentiment": "Negative",
            "aspect": "price",
            "target_entity": "LOCATION1"
        }
    ],
    "id": 1430,
    "text": "    LOCATION1 is transforming and the prices will go up and up"
},

{
    "opinions": [
        {
            "sentiment": "Positive",
            "aspect": "live",
            "target_entity": "LOCATION1"
        },
        {
            "sentiment": "Negative",
            "aspect": "price",
            "target_entity": "LOCATION1"
        }
    ],
    "id": 127,
    "text": "  If I had the money, I would live near LOCATION1  "
},
```


FIGURE 2.2 – Un extrait du jeu de données Sentihood.

Dans le jeu de données Sentihood, une opinion est un tuple composé d'un emplacement, d'un aspect et d'un sentiment (figure 2.2). Les aspects dominants du jeu de données sont : « *price, safety, transit-location, et general* », les emplacements sont masqués par *location1* et *location2* et les sentiments sont *positive* et *negative*.

En effet, l'aspect *general* est l'aspect le plus fréquent avec plus de 2000 phrases ainsi que le sentiment le plus dominant est *positive*. Puisque chaque phrase peut contenir une ou plusieurs opinions, le nombre total d'opinions (5920) est supérieur au nombre de phrases dans cet ensemble de données.

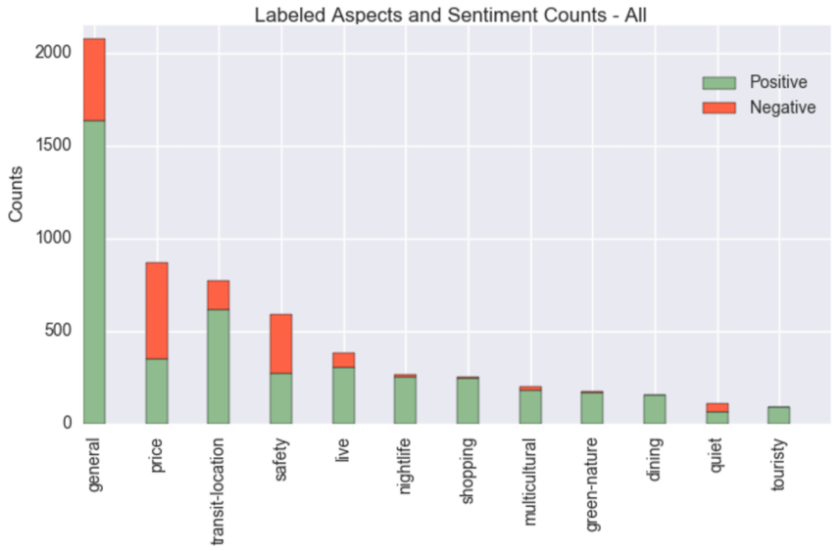

FIGURE 2.3 – Les statistiques des aspects et des sentiments présents dans le jeu de données Sentihood (Saeidi *et al.*, 2016).

Pour collecter les données, le contenu de chaque paire question-réponse de la plateforme Yahoo a été agrégé et divisé en phrases. Saeidi *et al.* (2016) n'ont conservé que les phrases qui mentionnent un emplacement et ils ont supprimé le reste. Dans une seule phrase, les auteurs ont trouvé que le nombre d'emplacements varie entre 1 à 50. Ainsi, ils ont sélectionné uniquement les phrases contenant une ou deux mentions d'emplacements. D'ailleurs, les noms des emplacements ont été extraits à l'aide du répertoire GeoNames et limités à ceux situés à l'intérieur de Londres.

Dans ces conditions, trois annotateurs humains ont été sélectionnés pour cette tâche. Ils ont utilisé l'outil d'annotation BRAT (Stenetorp *et al.*, 2012) pour simplifier la tâche d'annotation. Une liste prédéfinie de 11 aspects a été proposée aux annotateurs « *live, safety, price, quiet, dining, nightlife, transit-location, touristy, shopping, green-culture and multicultural* ». En plus, l'aspect *general* a été proposé pour faire référence à une opinion générique.

Pour chaque aspect, les annotateurs devaient attribuer un sentiment, ils n'ont fourni que des étiquettes des sentiments *positive* et *negative*. Ils ont éliminé l'étiquette *neutral*, car c'était rare de trouver des cas où les aspects sont discutés sans polarité. En ce qui concerne le processus d'annotation, les trois annotateurs ont annoté le jeu de données et comparé leurs versions. En cas de désaccord, le vote de la majorité était considéré comme l'annotation correcte. Ils ont attribué l'étiquette "Non pertinent" ou "Incertain" aux phrases qui ne sont pas clairement étiquetables.

À partir de la liste des 11 aspects proposée aux annotateurs, Saeidi *et al.* (2016) ont sélectionné les phrases qui contiennent les 4 aspects les plus fréquents à savoir ; *price*, *safety*, *transit-location* et *general*. À partir de cette sélection, les auteurs ont fixé les ensembles d'entraînement, de développement et de test, ayant chacun respectivement 70%, 10% et 20% des données. Pour tous les modèles que nous détaillerons dans le reste de ce mémoire, la même séparation de données a été adoptée.

2.3 Représentation de la tâche TABSA

Dans la tâche d'analyse des sentiments à base d'aspects ciblés (TABSA), la représentation d'une phrase doit contenir les informations nécessaires pour identifier l'aspect, le sentiment et l'entité cible qui est l'emplacement dans notre cas. En d'autres termes, la représentation doit refléter le contexte dans lequel l'emplacement est discuté.

Lorsqu'un humain effectue cette tâche, il se concentre de manière sélective sur les parties pertinentes de la phrase et acquiert les informations nécessaires pour construire une représentation interne de la cible dans son esprit. La figure 2.4 montre deux exemples de phrases avec deux emplacements. Le contexte est mis en évidence selon un humain avec l'accolade rouge pour *location1* et l'accolade bleu pour *location2*.

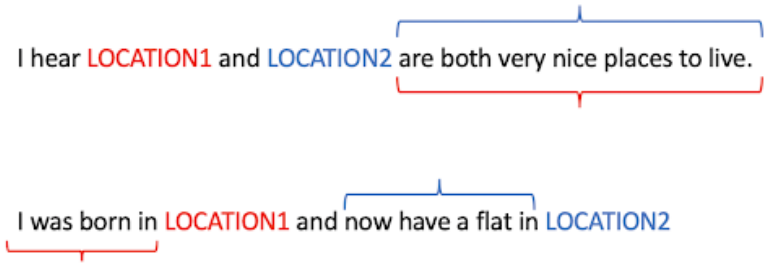

FIGURE 2.4 – Exemples de phrases à plusieurs emplacements.

Comme l'illustre la figure 2.4, deux emplacements partagent parfois le même contexte (la phrase du haut). Ils peuvent également avoir des contextes distincts apparaissant avant ou après la mention de l'emplacement (la phrase du bas).

Pour qu'un modèle identifie le sentiment d'un aspect relié à un emplacement spécifique, il est indispensable d'avoir une représentation concentrée sur le contexte approprié.

Nous pouvons définir la tâche TABSA comme un problème de classification à trois classes. Étant donné une phrase S, un ensemble d'emplacements T={*location1, location2*} et un ensemble fixe d'aspects A={*price, safety, transit-location, general*}, la tâche consiste à prédire le sentiment y $\in$ {*positive, negative, none*} sur l'ensemble complet des paires d'aspects cibles {(a, t) : a $\in$ A, t $\in$ T}. La classe none indique qu'une phrase ne contient pas d'opinion pour une paire donnée. Par exemple, la phrase "*location1 is very secure*" donne la paire (*location1, safety*) avec une polarité positive, tandis que (*location1, price*) n'a pas de polarité. Ainsi, la paire appartient à la classe *none*. Cette classe est ajoutée pour les paires inexistantes dans la phrase.

Saeidi *et al.* (2016) ont introduit la classe *none* dans le but d'étiqueter une paire d'aspects cibles si l'aspect est absent. En d'autres termes, une phrase sera convertie en quatre entrées pour représenter les quatre aspects sélectionnés par les auteurs. Le tableau 2.1 montre une phrase du jeu de données Sentihood (illustrée en rouge) et les nouvelles entrées pour représenter le reste des aspects.

id	**Phrase**	**Étiquettes**
1000	location1 is one of the most expensive areas of london location.	(location1, price, negative)
1000	location1 is one of the most expensive areas of london location.	(location1, general, none)
1000	location1 is one of the most expensive areas of london location.	(location1, transit-location, none)
1000	location1 is one of the most expensive areas of london location.	(location1, safety, none)

TABLEAU 2.1 – Exemple d'une phrase d'entrée et des étiquettes de sortie.

La figure 2.5 montre les statistiques des classes et des emplacements cibles pour les ensembles d'entraînement, de test et de développement après l'ajout de la classe *none* au jeu de données Sentihood.

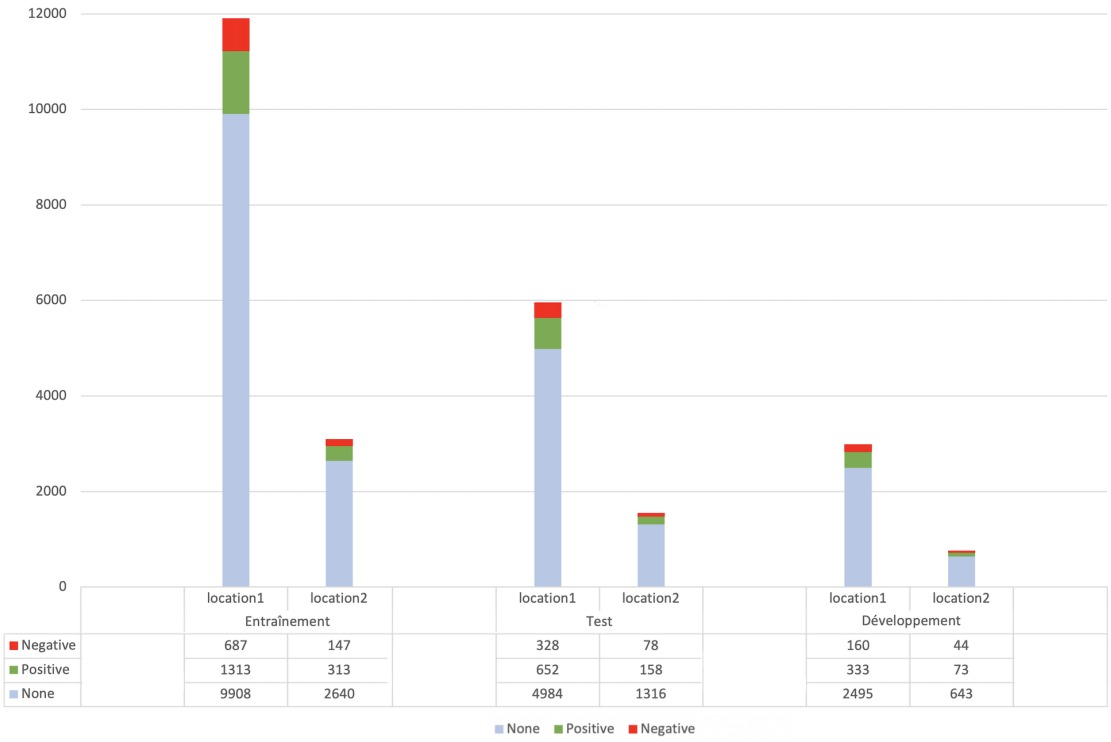

FIGURE 2.5 – La distribution des emplacements et des classes pour les ensembles d'entraînement, de test et de développement.

2.4 TABSA basée sur les réseaux récurrents

La tâche d'analyse des sentiments à base d'aspects ciblés peut être traitée avec des modèles basés sur les réseaux récurrents. Dans ce qui suit, nous présentons les travaux effectués sur la tâche TABSA en utilisant le jeu de données Sentihood.

2.4.1 Les modèles LSTM de Saeidi et al. (2016)

Saeidi *et al.* (2016) ont proposé deux modèles de base avec le jeu de données Sentihood. Un premier modèle basé sur la régression logistique (LR) et un deuxième modèle basé sur le LSTM. En ce qui suit, nous analyserons uniquement le modèle basé sur LSTM.

Le LSTM utilisé par Saeidi *et al.* (2016) peut être considéré comme une adaptation du modèle TDLSTM (Dong *et al.*, 2014). TDLSTM est un modèle qui permet d'encoder la structure

séquentielle d'une phrase et représenter une cible donnée en utilisant un vecteur moyenné sur les sorties cachées des instances cibles (Ma *et al.*, 2018). Ce modèle LSTM utilise donc les sorties cachées à la position des emplacements (*location1, location2*) en supposant que tous les emplacements ont le même degré d'importance dans une phrase.

Comme mentionné dans la section 2.3, Saeidi *et al.* (2016) ont traité la tâche d'identification du sentiment comme une tâche de classification à trois classes. Ils ont défini une tâche de classification pour chacun des quatre aspects séparément. Aussi, ils ont utilisé une couche SoftMax pour calculer la probabilité des classes {positive, negative, none}, étant donné la représentation spécifique à un emplacement. La figure 2.6 présente un LSTM bidirectionnel qui donne une représentation pour chaque mot de la phrase, et calcule la probabilité du sentiment à l'aide de la couche SoftMax pour la sortie de chaque emplacement. Ce LSTM est entraîné pour identifier le sentiment de l'aspect « *price* » spécifiquement. Pour la phrase donnée en exemple, la sortie devrait être positive pour *location1* et negative pour *location2*.

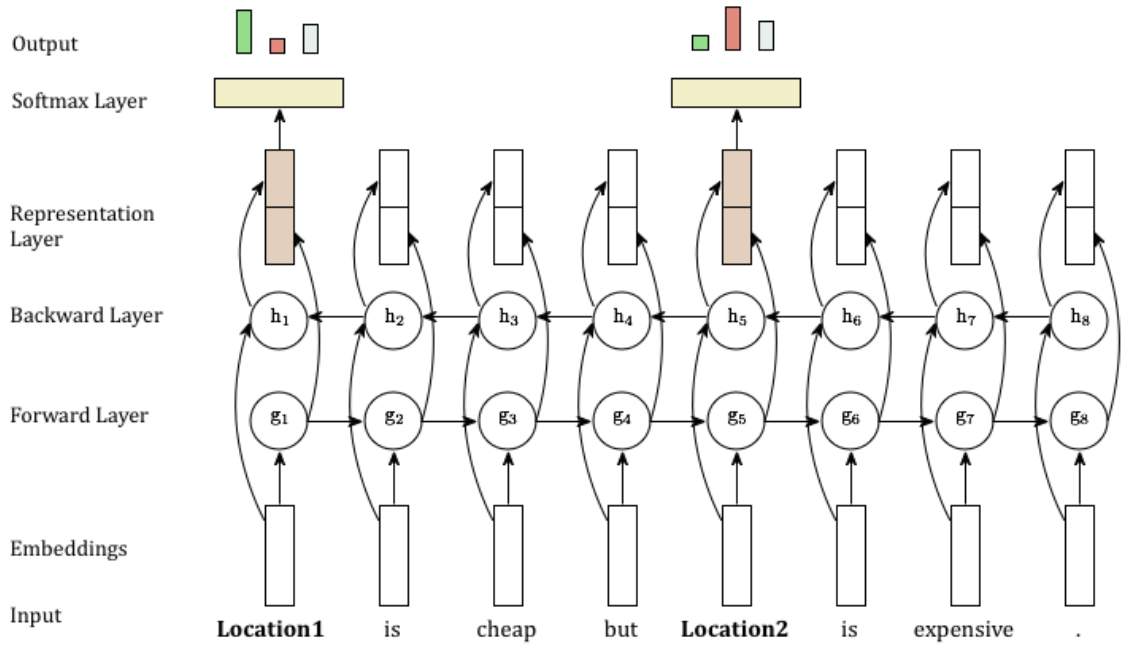

FIGURE 2.6 – Exemple d'un LSTM qui calcule la probabilité des sentiments au niveau de l'emplacement (Saeidi *et al.*, 2016).

Pour apprendre les représentations de chaque emplacement dans la phrase, les auteurs ont testé trois variantes du modèle LSTM (Saeidi *et al.*, 2016; Saeidi, 2017) :

- LSTM Unidirectionnel - Final (uni LSTM-Final) : pour chaque emplacement {*location1, location2*} dans la phrase d'entrée, une représentation finale du LSTM avant est introduite dans la couche SoftMax.
- LSTM Bidirectionnel - Final (biLSTM-Final) : pour chaque emplacement existant dans la phrase, une représentation finale du LSTM bidirectionnel est introduite dans la couche SoftMax. La représentation finale est obtenue en concaténant les deux vecteurs cachés avant et arrière).
- LSTM Bidirectionnel - Index (biLSTM-Index) : l'entrée de la couche SoftMax est une représentation partielle dans les deux sens (avant et arrière) jusqu'à l'index de l'emplacement dans la phrase. Dans la figure 2.6, l'index de *location2* est 4. Le vecteur avant tient compte donc de $\{g_1, g_2, g_3, g_4, g_5\}$, tandis que le vecteur arrière de $\{h_8, h_7, h_6, h_5\}$.

Puisque le jeu de données est déséquilibré (voir figure 2.5), Saeidi *et al.* (2016) ont entraîné le modèle LSTM par lot, où chaque lot représente 10% de l'ensemble d'entraînement échantillonné au hasard à partir de chacune de ces trois classes {*positive, negative, none*}. Ils ont optimisé les hyperparamètres en cherchant les valeurs optimales sur l'ensemble de développement. Le meilleur modèle qui possède la plus faible perte (loss) est obtenu lorsque le taux d'apprentissage est de 0.01, la taille des couches cachées est 50 et la taille du batch est 150.

Dans une phase d'évaluation donnée, le modèle détecte la présence des quatre aspects A={*price, safety, transit-location, general*}. Il retourne la classe *positive* ou *negative*, si la phrase implique la présence d'un aspect. Sinon, il retourne la classe *none*.

Pour évaluer la performance des modèles LSTM, les auteurs ont utilisé la F-mesure macro-moyenne et l'AUC pour la détection d'aspects, tout en ignorant la classe none. Ils ont aussi utilisé l'accuracy et l'AUC macro-moyenne pour la classification des sentiments. Les résultats sont présentés dans le tableau 2.2 :

Modèle	Aspect		Sentiment	
	F-Mesure	AUC	Accuracy	AUC
uniLSTM-Final	26.8	49.7	37.8	47.9
biLSTM-Index	69.3	89.7	81.9	83.9
biLSTM-Final	68.9	89.8	82.0	85.4

TABLEAU 2.2 – Les résultats des modèles LSTM sur le jeu de données Sentihood.

(Saeidi *et al.*, 2016; Saeidi, 2017)

2.4.2 Sentic-LSTM de Ma et al. (2018)

Ma *et al.* (2018) proposent le modèle Sentic-LSTM qui permet d'intégrer la base de connaissances SenticNet (Cambria *et al.*, 2018) dans un LSTM. L'architecture du modèle se compose principalement de deux composantes : un encodeur de séquence et un module d'attention hiérarchique.

L'encodeur de séquence se base sur un LSTM bidirectionnel et intègre la base de connaissances SenticNet qui contient 100 000 concepts de langage naturel. Cette base de connaissances fournit non seulement une représentation au niveau du concept, mais également des liens sémantiques vers les aspects et leurs sentiments. L'encodeur de séquence est capable d'intégrer cette base de haute dimensionnalité grâce à AffectiveSpace (Cambria et Hussain, 2015) qui intègre les concepts de SenticNet à des plongements continus de faible dimension sans perdre la relation sémantique et affective d'origine. Ma *et al.* (2018) ont pu introduire AffectiveSpace à l'aide d'une matrice de transformation qui permet de mapper les nouveaux concepts. Ils ont également ajouté une porte de sortie qui filtre les connaissances affectives. L'encodeur de séquence transforme les plongements de mots en une séquence de sorties cachées (les sorties du LSTM). Au-dessus des sorties cachées, deux types de mécanismes d'attention sont construits : un attention au niveau de la cible (target) et un attention au niveau de la phrase (figure 2.7).

L'attention au niveau de la cible prend comme entrée les sorties cachées aux positions de l'expression cible et calcule un vecteur d'attention sur ces mots. Le résultat de la composante d'attention au niveau de la cible est une représentation de la cible. Ensuite, la représentation de la cible est utilisée pour calculer un deuxième vecteur d'attention au niveau de la phrase. Le mécanisme d'attention au niveau de la phrase renvoie un vecteur pour chaque paire aspect cible (a, t) de la phrase (figure 2.7).

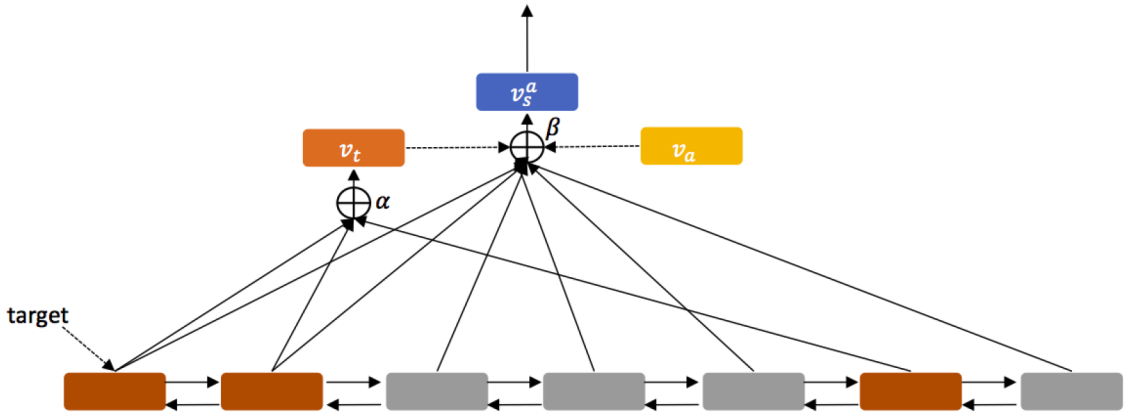

FIGURE 2.7 – L'architecture du modèle Sentic-LSTM (Ma *et al.*, 2018).

α et β représentent respectivement les vecteurs d'attention au niveau de la cible et de la phrase. v_a et v_t représentent respectivement le plongement d'aspects a et la représentation vectorielle de la cible. v_s^a est une combinaison linéaire de v_a et v_t qui est introduite dans un classifieur à classes multiples pour résoudre la polarité.

Pour la tâche TABSA, Ma *et al.* (2018) ont utilisé l'accuracy moyenne sur les aspects pour la classification des sentiments. Aussi, ils ont rapporté l'accuracy stricte et la F-mesure macro-moyenne pour la détection d'aspects tout en ignorant la classe *none*. À titre de rappel, la classe *none* est utilisé lorsqu'aucun aspect n'est trouvé dans une phrase. Les auteurs ont évalué la détection d'aspects comme un problème de classification multi-labels pour que les résultats soient moyennés sur des emplacements plutôt que sur des aspects.

Pour un jeu de données D, les métriques de la classification multi-label sont définies par les équations 2.1 et 2.2 :

$$Accuracy_{Strict} = \frac{1}{|D|} \sum_{t \in D} \sigma(Y_t = \widehat{Y}_t) \tag{2.1}$$

$$\text{F-Mesure}_{macro} = 2 \frac{pr\acute{e}cision_{macro} rappel_{macro}}{pr\acute{e}cision_{macro} + rappel_{macro}} \tag{2.2}$$

Où :

$|D|$ est le nombre de cibles dans D.

t représente l'emplacement dans une phrase.

Y_t représente l'ensemble des étiquettes réelles à associer à l'entité t.

$\widehat{Y_t}$ représente l'ensemble des étiquettes prédites pour l'entité t.

σ est la fonction qui indique si $\widehat{Y_t}$ correspond à Y_t.

La F-Mesure$_{macro}$ est basée sur la $précision_{macro}$ et le $rappel_{macro}$ qui sont détaillées respectivement par les équations 2.3 et 2.4.

$$précision_{macro} = \frac{1}{|D|} \sum_{t \in D} \frac{|Y_t \cap \widehat{Y_t}|}{\widehat{Y_t}} \tag{2.3}$$

$$rappel_{macro} = \frac{1}{|D|} \sum_{t \in D} \frac{|Y_t \cap \widehat{Y_t}|}{Y_t} \tag{2.4}$$

Les résultats du modèle Sentic-LSTM sont présentés dans le tableau 2.3.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
Sentic-LSTM	67.4	78.2	—	89.3	—

Tableau 2.3 – Les résultats de Sentic-LSTM de Ma *et al.* (2018) sur le jeu de données Sentihood.

Le symbole "—" signifie non reporté.

2.4.3 Le Réseau d'entités récurrentes de Liu et al. (2018)

Liu *et al.* (2018) ont proposé l'utilisation des réseaux d'entités récurrentes (EntNet). C'est un modèle de réseau de neurones équipé d'une mémoire dynamique à long terme qui permet de maintenir et de mettre à jour une représentation dès qu'il y a de nouvelles informations (Henaff *et al.*, 2016). Le EntNet peut être considéré comme un ensemble de RNNs qui partagent les mêmes paramètres, dont les slots mémoires sauvegardent les états cachés. Le modèle est conçu pour traiter les données sous forme séquentielle et se compose de trois modules : un encodeur d'entrée, une mémoire dynamique et un module de sortie.

Les modèles EntNet et LSTM se ressemblent dans leurs mécanismes de portes qui leur permettent de changer ou modifier l'information sauvegardée dans l'état caché. Toutefois, EntNet se distingue par la représentation des informations dans la mémoire. Le modèle LSTM utilise un seul vecteur mémoire, où les éléments peuvent interagir. Tandis que le modèle EntNet possède plusieurs chaînes mémoires indépendantes et avec peu d'interactions entre elles. L'indépendance des chaînes mémoires dans EntNet permet non seulement de capturer les dépendances à longue portée, mais aussi de sauvegarder les informations de manière plus stable (Liu *et al.*, 2018).

Le modèle EntNet représente et sauvegarde les entités dans une mémoire de taille fixe. L'état caché de chaque entité est suivi par une cellule mémoire équipée d'une clé qui indique l'identité de l'entité. Lors de la lecture d'une phrase, les chaînes mémoires maintiennent les représentations des entités à jour. Cette opération de mise à jour est contrôlée par un mécanisme de porte, qui surveille la similitude entre l'entrée actuelle et les chaînes mémoires, et décide ensuite de mettre à jour une ou plusieurs chaînes mémoires.

Liu *et al.* (2018) ont créé un modèle EntNet bidirectionnel auquel ils ont ajouté un mécanisme de mise à jour retardée de la mémoire (en anglais ; Delayed Memory Update, d'où l'acronymie Dmu). Le mécanisme du modèle Dmu-Entnet permet de suivre et de mettre à jour les états des entités au bon moment avec la mémoire externe. Plus précisément, Dmu-Entnet (figure 2.8) gère un certain nombre de chaînes mémoire h^j, une pour chaque emplacement avec la clé k^j. Lors de la progression de la phrase, les états h^j sont dynamiquement mis à jour en utilisant la récurrence de retard d^j et en tenant compte des activations précédentes.

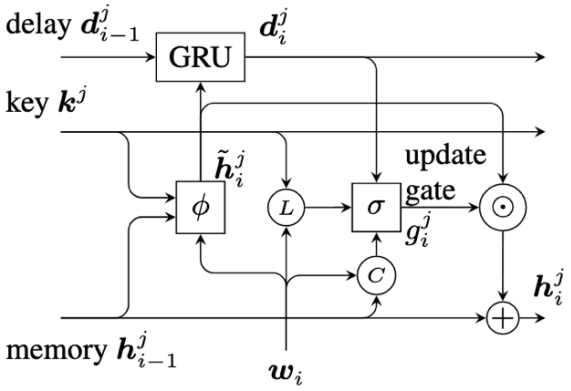

FIGURE 2.8 – La chaîne mémoire du modèle Dmu-Entnet à l'instant i.

σ, ϕ et GRU représentent respectivement la fonction d'activation sigmoïde, la fonction ReLU et le mécanisme de porte GRU (Chung *et al.*, 2014). Les nœuds entourés L, C, $\odot$ et $\oplus$ représentent respectivement l'emplacement, les termes du contenu, le produit Hadamard et l'addition.

La figure 2.8 présente une chaîne mémoire du modèle Dmu-Entnet. À l'instant i, la représentation d'entrée w_i est introduite dans la j^e chaîne mémoire h^j via un mécanisme de déclenchement g^j qui contrôle les mises à jour dans cette chaîne mémoire. Le mécanisme de déclenchement contient trois composants : le contenu C, l'emplacement L et le délai. Ces trois composants peuvent conduire à l'activation de g_i^j, mais ils diffèrent par la façon dont ils activent la porte :

- Le contenu $C = w_i \cdot \vec{h}_{i-1}^j$ déclenche l'activation de la porte lorsque le contenu des entités $\vec{h}_{i-1}^j$ correspond à l'entrée.
- L'emplacement $L = w_i \cdot k^j$ déclenche l'activation de la porte lorsque la clé k^j correspond à l'entrée.
- Le délai $\vec{v} \cdot \vec{d_i^j}$ modélise comment et quand la porte a été activée dans le passé et comment les activations passées devraient influencer l'actuel.

La porte de mise à jour retardée est donc définie comme suit :

$$\overrightarrow{g}_i^j = \sigma(w_i . \overrightarrow{h}_{i-1}^j + w_i . k^j + \overrightarrow{v} . \overrightarrow{d}_i^j) \tag{2.5}$$

Où :

$\overrightarrow{g}_i^j$ est la valeur de la porte de mise à jour pour la j^{e} mémoire à l'instant i,

$\overrightarrow{h}_{i-1}^j$ est la représentation responsable de suivre l'état de la j^{e} entité (contenu),

k^j est le plongement de l'entité suivie,

$\overrightarrow{d}_i^j$ est la récurrence du retard (délai),

$\overrightarrow{v}$ est un vecteur de paramètres.

Une fois que la valeur de la porte de mise à jour retardée est calculée, la mémoire est mise à jour comme suit :

$$\overrightarrow{\circ}_i^j = \overrightarrow{h}_{i-1}^j + \overrightarrow{g}_i^j \odot \overrightarrow{\tilde{h}}_i^j \tag{2.6}$$

Où :

$\overrightarrow{\circ}_i^j$ est la représentation de la mémoire non normalisée,

$\overrightarrow{\tilde{h}}_i^j$ est le nouveau vecteur mémoire candidat.

Le modèle prédit la polarité du sentiment $\hat{y}$ pour le plongement de la cible t et le plongement de l'aspect a en incorporant les états de toutes les entités suivies sous la forme d'une somme pondérée u :

$$p^j = softmax\left((k^j)^\top W_{att} \begin{bmatrix} t \\ a \end{bmatrix}\right) \tag{2.7}$$

$$u = \sum_j p^j h_m^j \tag{2.8}$$

$$\hat{y} = softmax(R\phi(Hu + a)) \tag{2.9}$$

Où :

W_{att} est une matrice de poids entraînable,

p^j est la valeur de l'entité suivie,

m est la longueur de la phrase.

L'entraînement est effectué en utilisant la perte de l'entropie croisée :

$$\mathcal{L} = CrossEntropy(y, \hat{y}) \tag{2.10}$$

Pour récapituler, Liu *et al.* (2018) ont apporté trois améliorations au modèle EntNet de Henaff *et al.* (2016). Premièrement, ils ont modélisé le délai d'activation des portes de mise à jour g^j avec le GRU. Deuxièmement, ils ont incorporé la cible et l'aspect au moment de la détermination de l'attention avec la fonction Softmax. Troisièmement, le nouveau modèle proposé est bidirectionnel.

Liu *et al.* (2018) ont initialisé leur modèle avec GloVe (300-D, entraîné sur 42 milliards de jetons et 1.9 millions de vocabulaire). L'entraînement du modèle Dmu-Entnet est effectué sur 800 époques avec l'optimiseur FTRL (McMahan *et al.*, 2013), la taille du lot de 128 et le taux d'apprentissage de 0.05. Une fonction d'abandon est appliquée à la sortie avec un taux de 0.2.

Liu *et al.* (2018) ont traité la tâche d'identification du sentiment comme une tâche de classification à trois classes. Le modèle prédit la classe du sentiment pour chaque paire aspect cible (a, t) en incorporant les états cachés de toutes les entités suivies sous la forme d'une somme pondérée.

Les auteurs ont rapporté la F-mesure macro-moyenne et l'AUC pour la détection d'aspects tout en ignorant la classe none, et ils ont utilisé l'accuracy et l'AUC macro-moyenne pour la classification des sentiments. Comme Ma *et al.* (2018), ils ont rapporté l'accuracy stricte pour la détection d'aspects. Les résultats des modèles EntNet et Dmu-Entnet, entraînés et testés sur le jeu de données Sentihood, sont présentés dans le tableau 2.4.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
EntNet	66.3	69.8	89.5	87.6	89.7
Dmu-Entnet	73.5	78.5	94.4	91.0	94.8

TABLEAU 2.4 – Les résultats des modèles de Liu *et al.* (2018) sur le jeu de données Sentihood.

2.5 TABSA basée sur l'apprentissage par transfert

Dans ce qui suit, nous présentons les travaux basés sur l'apprentissage par transfert, réalisés sur la tâche TABSA en utilisant le jeu de données Sentihood.

2.5.1 Apprentissage par transfert

Avec le terme apprentissage par transfert, nous faisons référence à la technologie qui nous permet d'utiliser les connaissances acquises par l'entraînement d'un modèle pour une tâche, afin de résoudre une autre tâche connexe (voir figure 2.9). De cette manière, il est possible d'économiser du temps et des efforts de calcul.

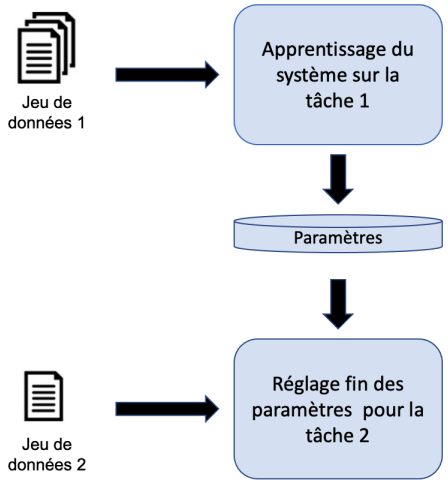

FIGURE 2.9 – Le mécanisme d'apprentissage par transfert.

Comme le montre la figure 2.9, l'apprentissage est divisé en deux phases : la première phase consiste à obtenir un modèle qui a été entraîné pendant longtemps sur un grand ensemble de GPU. La deuxième phase consiste à faire l'ajustement fin (ou fine-tuning) en utilisant un nouvel ensemble de données plus spécifique et plus petit. Les poids du modèle préentraîné à la première phase seront utilisés pour initialiser le nouveau modèle en fonction de la tâche à résoudre.

2.5.2 Le modèle BERT

Dans un contexte où l'apprentissage par transfert gagne de plus en plus de terrain, les modèles de plongement de mots ont eu de moins en moins d'importance avec l'apparition de modèles «contextualisés» tels que BERT (Devlin *et al.*, 2019). BERT signifie «Bidirectional Encoder Representations from Transformers», et ce dernier est capable de produire des représentations de mots en considérant le contexte de la phrase. Ce modèle de langage préentraîné a atteint de bons résultats en performance, néanmoins, la complexité de son architecture génère un coût d'entraînement plus important que des approches considérées plus classiques telles que Word2vec (Mikolov *et al.*, 2013a), Glove (Pennington *et al.*, 2014) et fastText (Mikolov *et al.*, 2017). D'ailleurs, la tendance générale est d'utiliser une version préentraînée du modèle plutôt que de l'entraîner à partir de zéro.

Aujourd'hui, le modèle BERT est devenu le choix le plus connu dans plusieurs tâches NLP. Nous trouvons aussi bien des versions préentraînées pour le domaine général, que pour des domaines spécialisés tels que SciBERT (Beltagy *et al.*, 2019) pour le domaine scientifique et BioBERT (Lee *et al.*, 2020) pour le domaine médical. Une simple procédure standard permet de spécialiser la version générale de BERT. Elle consiste à raffiner BERT-général sur un jeu de données spécialisé, tel le cas de Sun *et al.* (2019a) qui ont spécialisé BERT-général avec le jeu de données Sentihood. Cette approche de spécialisation du modèle général améliore considérablement la performance du modèle.

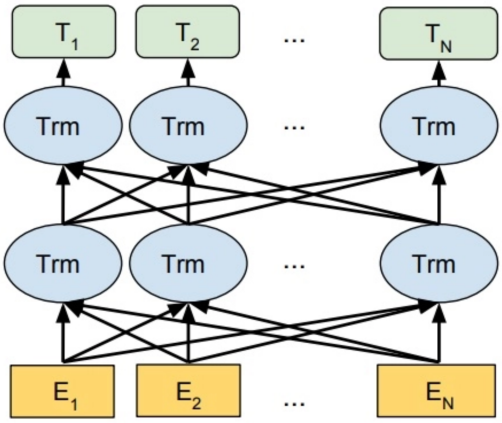

FIGURE 2.10 – L'architecture générale de BERT (Devlin *et al.*, 2019).

E_i : entrée, Trm : Transformer, T_i Sortie du modèle.

BERT est un réseau de neurones qui superpose une succession de couches Transformer (Vaswani *et al.*, 2017) pour créer des représentations contextualisées (voir figure 2.10). Le modèle de base se compose d'une couche d'intégration, d'un encodeur à 12 couches et d'une couche de regroupement (Devlin *et al.*, 2019). L'entraînement du modèle a été effectué sur deux tâches pour obtenir un modèle de langage préentraîné : une tâche de modélisation du langage masqué (Masked Language Modelling - MLM) et une tâche de prédiction de la phrase suivante (Next Sentence Prediction - NSP).

Les modèles de langage préentraînés fournissent un contexte aux mots, qui ont précédemment appris l'occurrence et les représentations de mots à partir de données d'apprentissage non annotées. BERT est un modèle de langage préentraîné basé sur le transfert de connaissances et conçu pour considérer le contexte d'un mot à la fois à gauche et à droite. Bien que ce concept soit simple, il a amélioré les résultats de plusieurs tâches de NLP telles que l'analyse des sentiments et les systèmes de questions et réponses (Devlin *et al.*, 2019).

Plusieurs facteurs doivent être pris en considération afin d'adapter BERT à une tâche spécifique (Devlin *et al.*, 2019). Parmi ces facteurs nous citons :

- La longueur de l'entrée (puisque la longueur maximale de BERT est de 512 jetons).
- La sélection de la dernière couche selon la tâche à accomplir.
- Le réglage fin du taux d'apprentissage.

2.5.2.1 Représentation de l'entrée

Le vocabulaire de BERT se constitue de 30 000 mots et sous-mots. C'est un mélange de plongement de WordPieces (Devlin *et al.*, 2019). Afin de remédier au problème de mots hors vocabulaire, BERT découpe chaque mot inconnu en un ensemble de sous-mots connus et existants dans le vocabulaire qui construisent les jetons à l'aide de la tokenisation WordPieces. Par exemple, le mot 'playing' est inconnu par le vocabulaire, donc la tokenisation de morceaux de mots le découpe en sous-mots 'play' et '##ing'.

Pour préentraîner BERT, les phrases d'entrée sont d'abord traitées par la méthode de tokenisation de morceaux de mots (Devlin *et al.*, 2019). Par la suite, les phrases sont découpées en mots ou en sous-mots avec un jeton spécial de classification [CLS] qui est ajouté au début de la séquence et un jeton spécial de séparation [SEP] pour marquer la fin de la phrase. La sortie finale du processus de tokenisation est une séquence d'entiers [1] qui correspond au vocabulaire de BERT. Si le mot ne se trouve pas dans le vocabulaire, un jeton spécial [UNK] est utilisé.

Cet ensemble de jetons est ensuite traité par trois différentes couches d'intégration avec les mêmes dimensions qui sont ensuite additionnées et transmises à la couche de l'encodeur. Chaque élément de la phrase d'entrée passera dans la couche du plongement de jeton (token embedding) pour obtenir son vecteur de représentation. Le plongement de position (position embedding) garde la notion de position des mots dans la phrase, tandis que le plongement de segment (segment embedding) garde la notion de position des phrases qui permet de distinguer les mots issus de chacune des phrases d'entrée. La figure 2.11 montre la somme des plongements de jeton, de segment et de position, qui est la représentation de l'entrée de BERT (Devlin *et al.*, 2019).

1. La séquence d'entiers représente les numéro de ligne dans la matrice de plongement de mots de BERT.

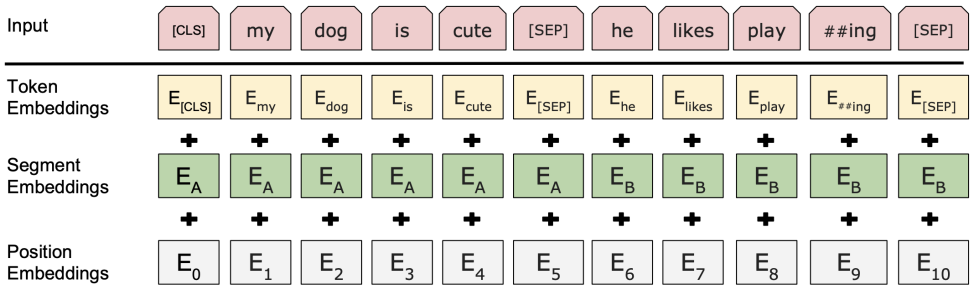

FIGURE 2.11 – La représentation de l'entrée dans le modèle BERT (Devlin *et al.*, 2019).

Les phrases d'entrée sont donc transformées en vecteurs, où chacun représente un jeton de la phrase d'une façon indépendante des autres. Pour cette raison, un ensemble de couches Transformer sera utilisé pour créer une représentation de plus en plus contextualisée d'une façon itérative. Cette représentation des entrées sera calculée grâce aux couches d'attention *Multi-Head*.

Jusqu'ici, nous avons présenté l'architecture de BERT ainsi que la représentation de l'entrée. Dans ce qui suit, nous allons expliquer la phase de préentraînement, à travers laquelle le modèle BERT apprend à créer des plongements contextualisés. Puis, nous allons aborder la phase de spécialisation, où le modèle est raffiné pour une tâche spécifique.

2.5.2.2 Phase de préentraînement

Rappelons que le modèle BERT est entraîné sur deux tâches : la tâche du modèle de langage masqué (MLM) et de la tâche de prédiction de la phrase suivante (NSP).

Le but de la tâche MLM est de masquer 15% des mots au hasard dans une phrase. Lorsque le modèle masque un mot, il le remplace par un jeton [MASK]. Le modèle apprend à prédire le mot masqué en utilisant le contexte à la fois à gauche et à droite du mot masqué à l'aide des Transformers. La figure 2.12 montre le fonctionnement de MLM, où le modèle choisit des mots au hasard et les remplace par le jeton spécial [MASK], par un autre mot du vocabulaire, ou bien il les conserve intacts.

En plus de l'extraction de contexte gauche et droite en utilisant MLM, BERT a un objectif clé supplémentaire qui diffère des travaux précédents, à savoir la prédiction de la phrase suivante. Le modèle apprend donc à prédire si B est vraiment une phrase qui suit A, étant donné une paire de phrases d'entrée A, B.

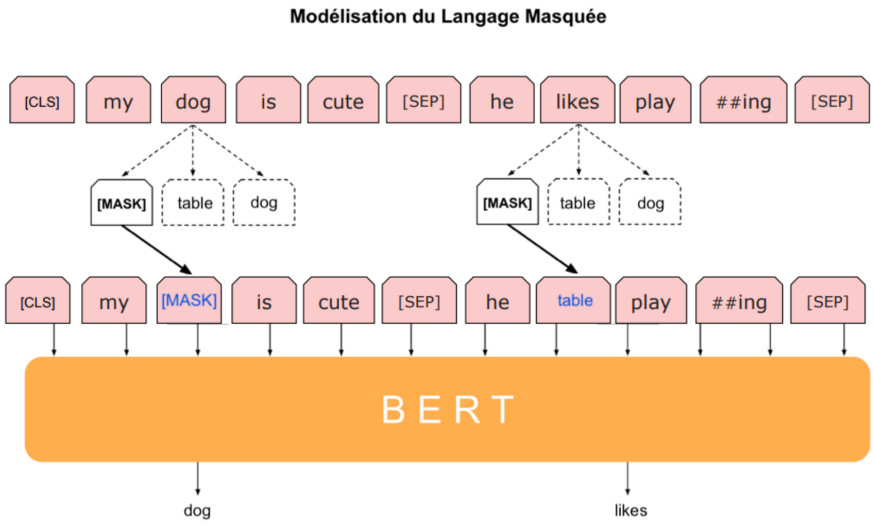

FIGURE 2.12 – Le fonctionnement du modèle de langage masqué (MLM).

Pour les deux phrases A et B, généralement, le préentraînement se fait en choisissant 50% du temps la phrase B qui suit la phrase A et lui donner une étiquette $IsNext$ et 50% du temps une phrase aléatoire du corpus et lui attribuer une étiquette $NotNext$ (voir tableau 2.5).

Phrase A	**Phrase B**	**Étiquettes**
my dog is cute	he likes playing.	$IsNext$
my dog is cute	penguins are flightless birds.	$NotNext$

TABLEAU 2.5 – Exemple d'une prédiction de la phrase suivante.

Devlin *et al.* (2019) ont préentraîné BERT-base et BERT-large sur les jeux de données BookCorpus (800 millions de mots) et English Wikipedia (2500 millions de mots) en commençant par la tokenisation de WordPiece, suivi par l'application de MLM et NSP. Pour le préentraînement, 40 itérations ont été exécutées sur le jeu de données de 3.3 milliards de mots. Les auteurs ont utilisé Adam (Kingma et Ba, 2014) avec un taux d'apprentissage de 1e-4, une probabilité d'abandon de 0.1 sur toutes les couches et une fonction d'activation GELU (Hendrycks et Gimpel, 2016) plutôt qu'une fonction RELU standard. La perte d'entraînement est la somme de la moyenne des probabilités de MLM et la moyenne des probabilités de NSP.

Le modèle BERT-base est constitué de 12 encodeurs de Transformer entraînées et 12 têtes d'attention. Chaque encodeur est composé de 768 couches cachées à propagation avant et le nombre total des paramètres préentraînés est 110M. Tandis que, BERT-large est constitué de 24 encodeurs, 16 têtes d'attention, 1024 couches cachées à propagation avant et de 340M paramètres.

Deux versions ont été proposées pour BERT-base et BERT-large : BERT-Uncased signifie que le texte a été mis en minuscules avec la suppression des accents (par exemple : *Zoë Baird* devient *zoe baird*), et BERT-Cased, signifie que nous préservons les majuscules et les accents. Généralement, le modèle BERT-Uncased est meilleur, sauf si la tâche est reliée à la reconnaissance d'entité nommée où les majuscules sont importantes.

2.5.2.3 Phase de spécialisation

La spécialisation de la tâche se fait en ajoutant une couche de sortie supplémentaire au-dessus du modèle BERT. Selon la tâche à accomplir, nous devons choisir le type de la dernière couche.

La figure 2.13 donne un aperçu de quelques tâches NLP dans lesquelles BERT a été utilisé. Les tâches de classification (a) et (b) s'effectue en ajoutant une couche de classification au-dessus de la sortie du Transformer. BERT peut être utilisé également pour la tâche de question/réponse (c) où le modèle reçoit une question et un paragraphe et doit en donner la réponse à partir du paragraphe. Dans ce cas, le modèle peut être entraîné en apprenant deux vecteurs supplémentaires qui marquent le début et la fin de la réponse dans le paragraphe fourni. De plus, le modèle BERT peut être utilisé pour la tâche de reconnaissance d'entité nommée (d) où il reçoit une série de mots dans un texte et doit marquer les différents types d'entités qui y apparaissent (Personne, Organisation, Date, etc.).

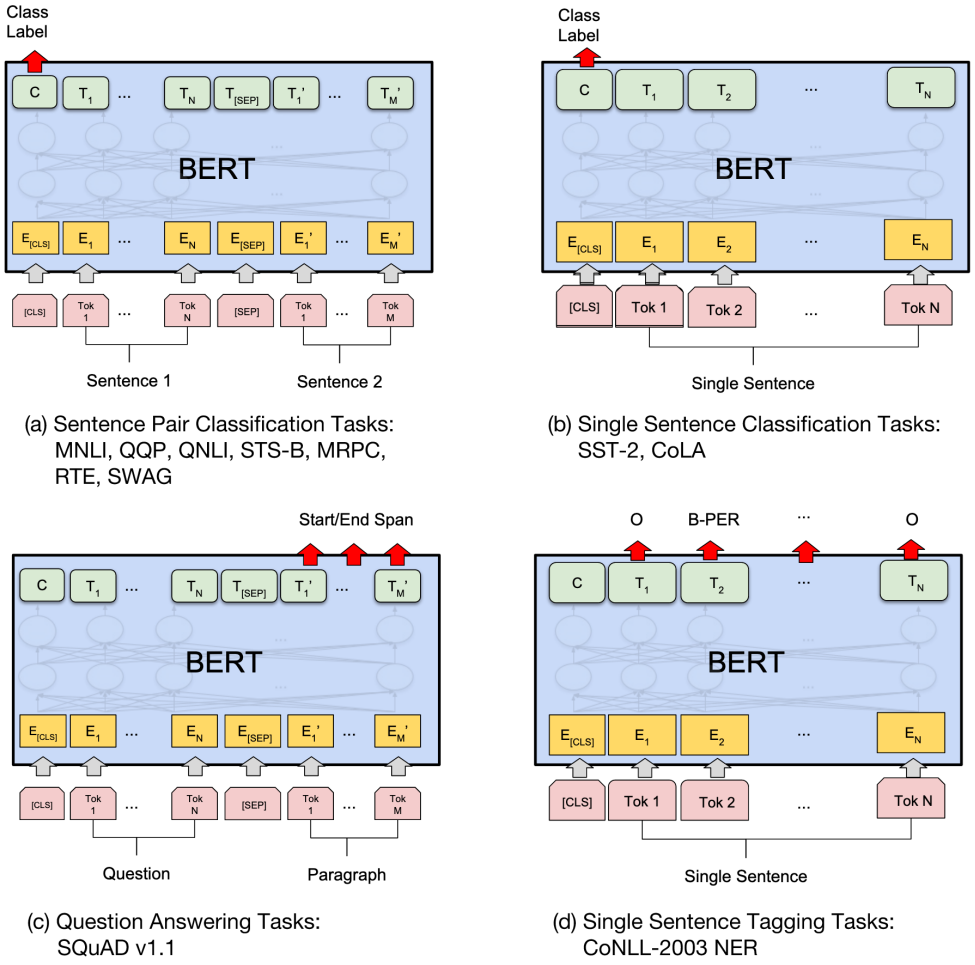

FIGURE 2.13 – Les tâches NLP réalisables par le modèle BERT (Devlin *et al.*, 2019).

Pour une tâche spécifique parmi celles décrites précédemment, les auteurs peuvent utiliser la partie BERT qui est déjà préentraîné et ajouter uniquement une couche linéaire pour la spécification de la tâche.

Lors de la spécialisation de la tâche sur le jeu de données Sentihood, Sun *et al.* (2019a) ont utilisé la tâche de classification (a) de la figure 2.13. Les auteurs ont utilisé le modèle BERT-base-Uncased sur le jeu de données SentiHood. Comme Devlin *et al.* (2019), les auteurs ont maintenu la probabilité d'abandon (dropout) à 0.1 sur l'ensemble des couches. Ils ont fixé le nombre d'itérations à 4, le taux d'apprentissage initial à 2e-5 et la taille du lot (batch) à 24.

Afin de construire une phrase auxiliaire à partir de la phrase initiale, Sun *et al.* (2019a) ont utilisé quatre méthodes que nous présenterons dans le tableau 2.6. Pour la méthode QA-M, ils ont généré une phrase sous forme d'une question. Pour la méthode NLI-M, ils ont généré une phrase simple. Pour la méthode QA-B, ils ont transformé TABSA en une tâche de classification binaire et ont créé trois phrases pour chaque paire (aspect, cible). Pour la méthode NLI-B, ils ont généré également trois pseudo-phrases.

Méthode	**Phrase A**	**Phrase B**	**Étiquettes**
QA-M	*location1* is often considered the safest area in town.	what do you think of the safety of *location1* ?	{*positive,negative, none*}
NLI-M	*location1* is often considered the safest area in town.	*location1 - safety*	{*positive,negative, none*}
QA-B	*location1* is often considered the safest area in town.	{ the polarity of the aspect safety of *location1* is positive , the polarity of the aspect safety of *location1* is negative , the polarity of the aspect safety of *location1* is none }	{*0, 1*}
NLI-B	*location1* is often considered the safest area in town.	{ location1 - safety - positive, location1 - safety - negative, location1 - safety - none }	{*0, 1*}

TABLEAU 2.6 – Les quatre méthodes utilisées pour générer une phrase auxiliaire.

Sun *et al.* (2019a) ont rapporté les mêmes mesures de performance que Liu *et al.* (2018) : L'accuracy stricte, la F-mesure et l'AUC macro-moyenne pour la détection d'aspects tout en ignorant la classe none ainsi que l'accuracy et l'AUC macro-moyenne pour la classification des sentiments. Les résultats sont présentés dans le tableau 2.7 :

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
BERT-pair-QA-M	79.4	86.4	97.0	93.6	96.4
BERT-pair-NLI-M	78.3	87.0	97.5	92.1	96.5
BERT-pair-QA-B	79.2	87.9	97.1	93.3	97.0
BERT-pair-NLI-B	79.8	87.5	96.6	92.8	96.9

TABLEAU 2.7 – Les résultats des modèles BERT-pair de Sun *et al.* (2019a).

2.6 Comparaison des résultats

Nous comparons dans cette section, les performances des modèles proposés pour la tâche TABSA et basés sur les réseaux de neurones récurrents, ainsi que les modèles basés sur l'apprentissage par transfert et détaillés dans les sections 2.4 2.5.

Saeidi *et al.* (2016) ont créé le jeu de données SentiHood pour la tâche TABSA. Ils ont proposé les modèles uniLSTM-Final, biLSTM-Index et biLSTM-Final. Ma *et al.* (2018) ont proposé une solution à la tâche TABSA en appliquant un mécanisme d'attention en deux étapes au niveau de la cible et de la phrase et en étendant le LSTM pour incorporer des informations supplémentaires. Inspiré par l'utilisation des modèles à mémoire augmentée dans le domaine de la traduction de langages, Liu *et al.* (2018) ont proposé l'utilisation des chaînes de mémoire externe avec un mécanisme de mise à jour retardée de la mémoire, permettant de suivre plusieurs entités cibles pour la tâche TABSA. Pour finir, Sun *et al.* (2019a) ont utilisé un modèle de langage préentraîné. Plus précisément, ils ont évalué BERT avec une paire de phrases d'entrées. Le tableau 2.8 présente les résultats de la détection des aspects et la classification des sentiments de tous les modèles cités ci-haut.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
uniLSTM-Final (Saeidi, 2017)	—	26.8	49.7	37.8	47.9
biLSTM-Index (Saeidi *et al.*, 2016)	—	69.3	89.7	81.9	83.9
biLSTM-Final (Saeidi *et al.*, 2016)	—	68.9	89.8	82.0	85.4
Sentic-LSTM (Ma *et al.*, 2018)	67.4	78.2	—	89.3	—
EntNet (Liu *et al.*, 2018)	66.3	69.8	89.5	87.6	89.7
Dmu-Entnet (Liu *et al.*, 2018)	73.5	78.5	94.4	91.0	94.8
BERT-pair-QA-M (Sun *et al.*, 2019a)	79.4	86.4	97.0	93.6	96.4
BERT-pair-NLI-M (Sun *et al.*, 2019a)	78.3	87.0	97.5	92.1	96.5
BERT-pair-QA-B (Sun *et al.*, 2019a)	79.2	87.9	97.1	93.3	97.0
BERT-pair-NLI-B (Sun *et al.*, 2019a)	79.8	87.5	96.6	92.8	96.9

TABLEAU 2.8 – Les résultats de la détection d'aspects et la classification des sentiments. Le symbole "—" signifie non reporté.

À partir des résultats du tableau 2.8, nous remarquons dans l'étude de Saeidi (2017) qu'une représentation LSTM avant uniquement (uniLSTM-Final) ne donne pas de bons résultats. Les performances des deux variations biLSTM donnent des résultats très similaires. Nous pouvons ainsi conclure que les performances sont plus élevées lorsque le contexte de la phrase est capturé dans les deux sens.

Quant aux résultats obtenus par Ma *et al.* (2018) sur le jeu de données SentiHood, nous notons que l'utilisation du modèle Sentic-LSTM a permis d'améliorer considérablement la détection des aspects et la classification des sentiments. La F-Mesure passe de 69.3% avec le modèle biLSTM-Index à 78.2% avec le modèle Sentic-LSTM. L'accuracy pour la détection d'aspects n'est pas rapporté par Saeidi *et al.* (2016) et l'AUC n'est pas rapporté par Ma *et al.* (2018).

En analysant les résultats de Liu *et al.* (2018), nous remarquons une amélioration des performances du modèle Dmu-Entnet par rapport à Sentic-LSTM de Ma *et al.* (2018) et par conséquent par rapport à celui de Saeidi *et al.* (2016). Nous rapportons une augmentation de l'accuracy de 67.4% à 73.5% entre le modèle Dmu-Entnet et le modèle Sentic-LSTM.

En analysant les résultats des modèles BERT-pair, nous remarquons que les 4 modèles surpassent les autres approches basées sur les réseaux de neurones récurrents par une marge substantielle. En plus de sa simplicité par rapport à Dmu-Entnet, le modèle BERT-pair-QA-B montre une amélioration de 2.3% pour l'accuracy et 2.2% pour l'AUC de la classification des sentiments.

En comparons les 4 modèles BERT-pair, nous concluons que le modèle BERT-pair-NLI-B offre les meilleures performances pour la détection d'aspects et que BERT-pair-QA-B offre les meilleures performances pour la classification des sentiments.

2.7 Conclusion

Au cours de ce chapitre, nous avons passé en revue les différents travaux de l'état de l'art traitant la tâche TABSA. D'abord, nous avons présenté le jeu de données Sentihood utilisé pour la tâche d'analyse des sentiments à base d'aspects ciblés. Ensuite, nous avons regroupé les travaux en deux catégories : les travaux basés sur les réseaux récurrents et les travaux basés sur l'apprentissage par transfert. Dans ce contexte, nous avons présenté les modèles uniLSTM-Final, biLSTM-Index et biLSTM-Final (Saeidi *et al.*, 2016; Saeidi, 2017), suivis par Sentic-LSTM (Ma *et al.*, 2018), Dmu-Entnet (Liu *et al.*, 2018) et BERT (Sun *et al.*, 2019a). Enfin, nous avons comparé les performances des modèles proposés.

Le chapitre suivant décrira notre contribution qui vise à apporter des modifications sur la phrase d'entrée et l'architecture du modèle BERT.

CHAPITRE III

APPROCHE PROPOSÉE

Au cours de ce chapitre, nous décrirons nos expérientations effectuées pour améliorer les performances des modèles de l'analyse des sentiments à base d'aspects ciblés (TABSA) sur le jeu de données Sentihood de Saeidi *et al.* (2016). Tout d'abord, nous présenterons notre méthodologie suivi par un modèle SVM qui servira comme base de comparaison. Comme le SVM ne représente pas le contexte et l'ordre des mots, sa performance limité en tant que modèle de base nous donne une idée de l'importance du contexte et de l'ordre des mots. Ensuite, nous présenterons notre modèle basé sur l'apprentissage par transfert : BERT-SentClass. Ce modèle est entraîné et testé sur le jeu de données Sentihood. Nous décrirons la modification de la phrase initiale pour préparer BERT à mieux performer dans la tâche TABSA. Nous détaillerons les réglages fins effectués sur notre modèle BERT-SentClass et les ajustements apportés à la nouvelle pseudo-phrase d'entrée. De plus, nous explorerons la possibilité de modifier la couche de sortie du modèle BERT-SentClass. Enfin, nous présenterons le modèle DistilBERT-SentClass.

3.1 Méthodologie

La tâche TABSA prédit le sentiment d'une cible par rapport à un aspect. Cette dernière est modélisée comme un problème de classification dans lequel un classificateur reçoit une phrase, un ensemble de cibles et un ensemble d'aspects puis prédit le sentiment *positive*, *negative* ou *none* pour l'ensemble complet des paires d'aspects cibles. Nous nous intéressons à la résolution de cette tâche sur le jeu de données Sentihood en proposant un modèle d'apprentissage automatique.

Nous avons exploré dans la revue de la littérature plusieurs modèles d'apprentissage automatique et notre comparaison des résultats révèle que les meilleures performances sur le jeu de données Sentihood ont été obtenues par le modèle BERT-pair grâce à la construction d'une phrase

auxiliaire artificielle. Ceci nous amène à explorer la possibilité de simplifier la tâche TABSA en regroupant toutes les informations nécessaires au modèle dans une seule phrase sans perte de performance. Notre approche consiste à insérer des balises d'une certaine manière dans la phrase d'entrée et à laisser le modèle BERT découvrir la tâche à apprendre.

Dans ce qui suit, nous allons explorer les possibilités de simplifier l'utilisation du modèle BERT en donnant une seule entrée au lieu d'une paire de phrases comme a fait Sun *et al.* (2019b). Notre défi est de trouver un moyen optimal de communiquer les informations nécessaires au modèle en une seule phrase d'entrée et d'analyser la performance du modèle par la suite.

3.2 Environnement d'entraînement

Nous avons effectué l'entraînement des modèles sur la machine virtuelle de Google communément connu sous le nom Colab [1]. Cette machine dispose d'une carte graphique Nvidia Tesla k80, TPU avec 180 Tflops, 35 Go de RAM et une capacité de stockage de 225 GB, ce qui permet un temps d'exécution optimal par rapport à notre environnement local. Les modèles ont été implémentés en utilisant les bibliothèques scikit-learn, Transformers et torch.

3.3 Le modèle de base

La tâche TABSA peut être considérée comme une tâche de classification, c'est pourquoi, nous pouvons utiliser les machines à vecteurs supports (SVM) ou la régression logistique (LR) en tant que modèle d'apprentissage automatique de base.

Dans le travail de Saeidi *et al.* (2016), les auteurs ont créé un modèle LR basé sur des caractéristiques linguistiques telles que les n-grammes et les étiquetages morpho-syntaxique (POS tagging). La prédiction du sentiment pour chaque aspect a été traitée comme une tâche distincte. Pour chaque emplacement présent dans la phrase (*location1*, *location2*), ils ont défini une représentation n-gramme pour la phrase d'entrée. Ils ont masqué l'emplacement cible avec un jeton spécial "$target\_loc$" afin de distinguer entre les emplacements s'il y en a plusieurs dans la même phrase.

1. https://colab.research.google.com

Dans ce qui suit, nous proposons SVM comme modèle de base avec une approche différente de celle de Saeidi *et al.* (2016). Durant le prétraitement, nous avons éliminé les mots vides et transformé le texte en minuscule. Une tokenisation a été effectuée permettant la séparation des phrases en jetons. Nous avons converti les jetons en vecteur d'entrée avec deux principales méthodes : les sacs de mots (BOW) et TF-IDF (Ahuja *et al.*, 2019). La figure 3.1 présente le processus de prétraitement du texte avant de passer les représentations vectorielles au modèle SVM.

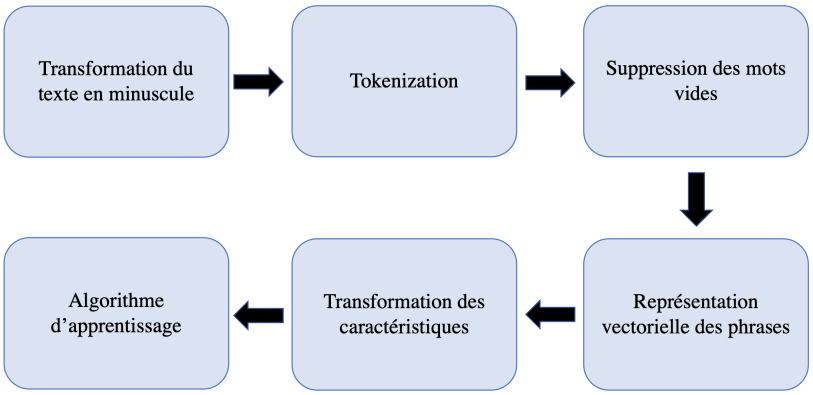

FIGURE 3.1 – Le processus de prétraitement et de transformation du texte.

La méthode BOW transforme le texte en vecteurs de longueur fixe en comptant simplement le nombre de fois qu'un mot apparaît dans un document.

La méthode de pondération TF-IDF permet d'évaluer l'importance d'un mot présent dans un document. Elle lui attribue un poids qui varie en fonction du nombre d'occurrences du mot dans le document et de la fréquence de ce mot dans le jeu de données (Belainine, 2017). Le TF-IDF est calculé comme suit :

$$\text{TF(fréquence du terme)} = \frac{\text{nombre d'apparition du mot dans un document}}{\text{nombre de termes dans un document}} \tag{3.1}$$

$$\text{TF-IDF(mot, document)} = \text{TF}(mot, document) * \text{IDF}(mot) \tag{3.2}$$

Pour implémenter les méthodes BOW et TF-IDF, nous avons utilisé la bibliothèque scikit-learn qui prend en entrée, toutes les phrases de notre jeu de données Sentihood et nous produit des

représentations vectorielles à l'aide des fonctions CountVectorizer [2] et TfidfVectorizer [3].

Prenons l'exemple des deux phrases suivantes :

- Phrase 1 : *"I would recommend location1."*
- Phrase 2 : *"location2 is an expensive area compared to location1."*

La figure 3.2 montre la représentation des deux phrases après l'utilisation de CountVectorizer et TfidfVectorizer en se servant du vocabulaire suivant déterminé à partir des deux phrases : ['area', 'compared', 'expensive', 'location1', 'location2', 'recommend'].

Count Vectorizer

	area	compared	expensive	location1	location2	recommend
Phrase 1	0	0	0	1	0	1
Phrase 2	1	1	1	1	1	0

TD-IDF Vectorizer

	area	compared	expensive	location1	location2	recommend
Phrase 1	0.000000	0.000000	0.000000	0.579739	0.000000	0.814802
Phrase 2	0.471078	0.471078	0.471078	0.335176	0.471078	0.000000

FIGURE 3.2 – La représentation avec CountVectorizer et TfidfVectorizer.

Nous avons utilisé l'implémentation du classificateur SVM de la bibliothèque scikit-learn. Nous avons effectué plusieurs exécutions sur l'ensemble de développement pour choisir la fonction de base radiale (rbf) comme noyau (kernel).

Pour le jeu de données Sentihood, nous avons intégré l'aspect après l'emplacement cible de la phrase. À titre d'exemple, pour l'aspect *general*, la phrase "*I would recommend location1.*" devient la pseudo-phrase "*I would recommend location1 general.*". Nous avons entrainé le modèle SVM-A qui représente l'exécution initiale sur le jeu de données Sentihood après intégration de l'aspect. Pour le modèle SVM-B, nous avons modifié l'ensemble d'entraînement sans toucher à l'ensemble de test. Le but de cette modification est d'équilibrer les données d'entraînement. Ainsi, pour chaque phrase de l'ensemble d'entraînement, nous avons supprimé les entrées de la classe *none* si au moins l'un des aspects y existe. Une phrase qui ne comporte aucun aspect est représentée par quatre entrées de classe *none*.

2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

3. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Le tableau 3.1 montre 8 entrées pour deux phrases distinctes. Dans le premier cas, nous gardons les quatre entrées et dans le deuxième nous supprimons les entrées de la classe *none* illustrées en rouge. Dans la figure 3.3, nous comparons les données d'entraînement avant et après la suppression sélective des entrées appartenant à la classe *none*.

ID	**Entrée**	**Aspect**	**Étiquette**
1008	location1 is the london station for coventry.	*general*	*none*
1008	location1 is the london station for coventry.	*price*	*none*
1008	location1 is the london station for coventry.	*safety*	*none*
1008	location1 is the london station for coventry.	*transit-location*	*none*
1000	location1 is one of the most expensive area.	*general*	*none*
1000	location1 is one of the most expensive area.	*price*	*negatif*
1000	location1 is one of the most expensive area.	*safety*	*none*
1000	location1 is one of the most expensive area.	*transit-location*	*none*

TABLEAU 3.1 – La suppression des entrées dans le but d'équilibrer les données d'entraînement.

En rouge : les entrées supprimées de l'ensemble d'entrainement.

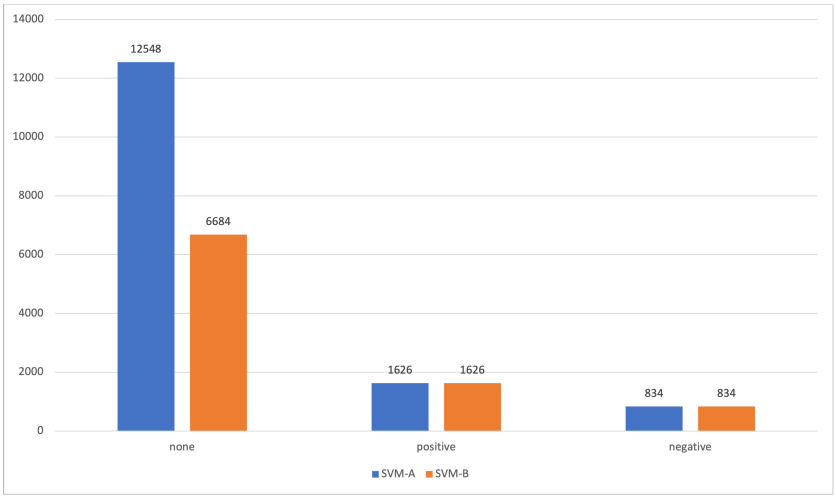

FIGURE 3.3 – La distribution de l'ensemble d'entraînement pour SVM-A et SVM-B.

Après avoir équilibré les données d'entraînement, nous avons effectué une deuxième modification (SVM-C) dans le but d'attacher l'aspect à l'emplacement cible. Reprenons l'exemple précédent "*I would recommend location1 general.*" qui devient "*I would recommend location1general.*". Cette deuxième modification donne lieu à un jeton qui reflète la cible d'un aspect. Les résultats des trois modèles SVM avec la pondération BOW et TF-IDF seront présentés dans le chapitre suivant.

3.4 Le modèle BERT-SentClass

Dans ce qui suit, Nous détaillerons les méthodes utilisées pour ajuster la phrase d'entrée. Nous présenterons le processus de réglage fin des hyperparamètres et la modification de la couche de sortie entièrement connectée.

3.4.1 La phrase d'entrée

Dans cette section, nous avons effectué des changements à la phrase d'entrée en y intégrant des éléments clés. La tâche TABSA peut être converti en un problème de classification à classes multiples pour prédire les classes {positive, negative, none}, ou binaire pour prédire les classes {0, 1}. Inspiré par Sun *et al.* (2019a), nous appellerons nos deux méthodes de classification, SentClass-M pour multiple et SentClass-B pour binaire.

Pour la méthode SentClass-M, nous avons sélectionné l'aspect, présent dans Sentihood, et nous l'avons intégré après l'emplacement dans la phrase initiale pour construire la pseudo-phrase qui va servir d'entrée à BERT telle que le montre la figure 3.4 :

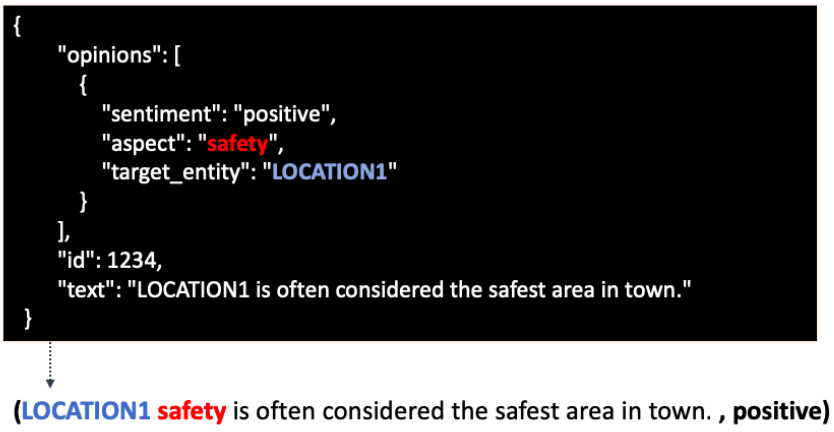

FIGURE 3.4 – La modification de la phrase d'entrée pour la méthode SentClass-M.

Pour la méthode SentClass-B, nous avons sélectionné l'aspect et le sentiment, présents dans le jeu de données Sentihood, et nous les avons intégrés dans l'ordre {*aspect*, *sentiment*} après l'emplacement cible telle que le montre la figure 3.5 :

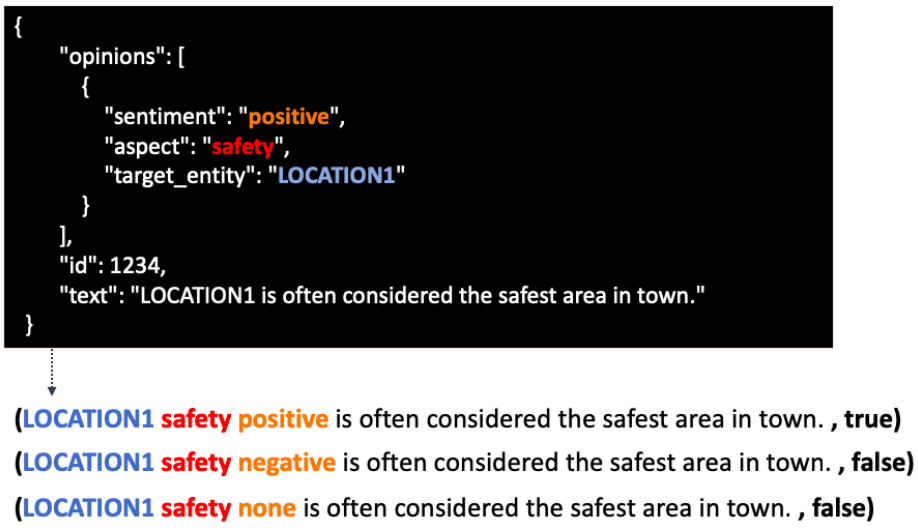

FIGURE 3.5 – La modification de la phrase d'entrée pour la méthode SentClass-B.

Dans la section suivante, nous allons procéder à un réglage fin des hyperparamètres afin de trouver une combinaison optimale.

3.4.2 Le processus de réglage fin

Dans cette section, nous explorons plusieurs façons pour peaufiner l'entraînement et améliorer les performances des modèles BERT-SentClass-M et BERT-SentClass-B pour la tâche TABSA en utilisant le jeu de données Sentihood.

Le réglage fin des hyperparamètres est une étape cruciale dans l'entraînement des modèles basés sur l'apprentissage profond. Devlin *et al.* (2019) recommandent de se concentrer sur le nombre d'epochs, la taille de lot, ainsi que le taux d'apprentissage pour améliorer la performance de BERT. L'évaluation des hyperparamètres a été effectuée sur l'ensemble de développement fourni par Saeidi *et al.* (2016) et qui represente 10% du jeu de données Sentihood.

Le nombre d'epochs correspond au nombre d'itérations où le modèle voit l'ensemble d'entraînement. Pour trouver le nombre d'epochs, nous allons exécuter les modèles BERT-SentClass-M et BERT-SentClass-B sur l'ensemble de développement avec la méthode *early stopping* qui interrompre l'entraînement du modèle lorsque la fonction de perte commence à augmenter et que les performances de généralisation se dégradent.

La taille de lot affecte la vitesse de convergence de l'optimisation de la fonction objectif et permet d'obtenir une meilleure performance de généralisation (Masters et Luschi, 2018). Nous allons tester nos modèles avec différentes tailles de lot {8, 16, 24, 32} sur l'ensemble de développement.

Le taux d'apprentissage permet de contrôler la vitesse avec laquelle le modèle va apprendre. Nous allons tester nos modèles sur l'ensemble de développement avec différentes taux d'apprentissage en considérant un choix aléatoire uniforme dans l'échelle logarithmique. Dans le but de trouver une configuration optimale d'hyperparamètres, nous allons opter pour la méthode *Random search* qui permet d'entraîner notre modèle avec des combinaisons aléatoires. Les résultats du processus de réglage fin seront présentés dans le chapitre suivant.

3.4.3 Les ajustements de la phrase d'entrée

Tout au long de nos expériences, nous avons apporté plusieurs ajustements supplémentaires à la phrase d'entrée. La figure 3.6 détaille les modifications apportées à la phrase "*Location1 is often considered the safest area in town.*", dont la construction a été expliquée dans la section 3.4.1.

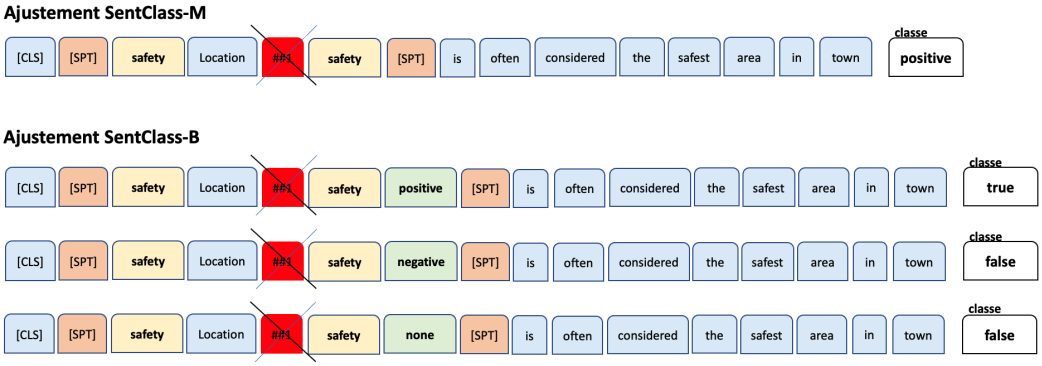

FIGURE 3.6 – Les ajustements de la phrase d'entrée pour BERT-SentClass-M et BERT-SentClass-B.

La première phrase de la figure 3.6 représente l'ajustement dans le cas multiple pour prédire les classes {positive, negative, none}. Nous avons encadré l'emplacement cible avec l'aspect et le jeton spécial [SPT]. Aussi, nous avons supprimé les sous-mots ('##1' et '##2') préalablement découpés à l'aide de la tokenisation des morceaux de mots 'WordPiece' de BERT (Devlin *et al.*, 2019). La deuxième partie de la figure 3.6 représente l'ajustement dans le cas binaire pour prédire les classes {0, 1}. Les résultats des ajustements de la phrase d'entrée seront présentés dans le quatrième chapitre.

3.4.4 La modification de la couche entièrement connectée

Lors de l'utilisation de BERT pré-entraîné, la spécialisation de la tâche de classification se fait en ajoutant une couche de sortie supplémentaire au-dessus du modèle. Sun *et al.* (2019a) ajoutent, dans leur cas, une seule couche linéaire. Dans nos expérimentations précédentes, nous avons gardé cette couche linéaire telle quelle pendant l'exécution de notre modèle BERT-SentClass. Dans cette section, nous avons utilisé deux couches linéaires au lieu d'une. La figure 3.7 montre la modification effectuée.

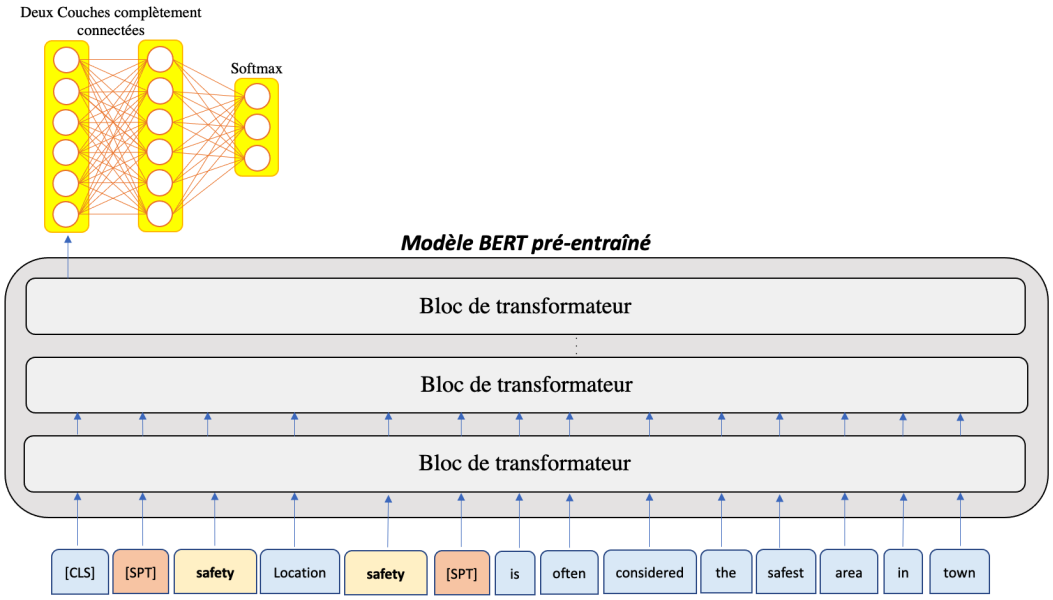

FIGURE 3.7 – La modification de la couche de sortie du modèle BERT-Sentclass.

3.5 Le Modèle DistilBERT

La distillation des connaissances (également appelée apprentissage enseignant-élève) est une technique de compression dans laquelle un petit modèle est entraîné à reproduire le comportement d'un modèle plus grand (Sun *et al.*, 2019c). L'élève est entraîné avec une distillation de perte sur les probabilités cibles de l'enseignant : $L_{ce} = \sum_i t_i * log(s_i)$ où t_i est une probabilité estimée par l'enseignant et s_i est une probabilité estimée par l'élève. L'objectif est de profiter de la distribution complète du modèle enseignant pour entraîner un modèle plus petit.

Après la publication de BERT, Sanh *et al.* (2019) ont créé une version distillée (approximative), qui est 40% plus petite et 60% plus rapide et utilise seulement la moitié du nombre de paramètres. Plus précisément, DistilBERT n'a pas de plongements de type jeton et ne conserve que la moitié des couches de BERT. En outre, cette version pré-entraînée du modèle BERT conserve 97% des capacités de compréhension de la langue.

En plus de la perte de la distillation (l_{ce}), DistilBERT a été entraîné avec la perte de la modélisation du langage masqué (l_{mlm}). Cela permet au modèle d'apprendre une représentation bidirectionnelle de la phrase. En effet, le modèle masque aléatoirement 15% des mots en entrée et les jetons masqués seront remplacés par le jeton spécial [MASK] dans 80% des cas, par un

autre mot du vocabulaire dans 10% des cas, et il les garde intact dans les 10% des cas restants.

Au total, les auteurs ont utilisé trois pertes d'entraînement (où Training losses) pour DistilBERT ; à savoir la perte de distillation, la perte de modélisation de langage masquée et la perte du plongement de cosinus afin d'aligner les directions des vecteurs des états cachés de l'élève et de l'enseignant. Cette triple perte garantit que le modèle étudiant DistilBERT apprend correctement de l'enseignant BERT et permet un transfert efficace des connaissances.

Pour la spécialisation de la tâche, nous avons utilisé le modèle DistilBERT-base$-Uncased$ sur le jeu de données Sentihood. Ce modèle est une version distillée du modèle BERT-base$-Uncased$. Les résultats du modèle DistilBERT-SentClass avec la pseudo-phrase d'entrée ajustée seront présentés dans le chapitre suivant.

3.6 Conclusion

Dans ce chapitre, nous avons proposé différentes étapes pour améliorer la performance des modèles d'apprentissage par transfert et pour simplifier la tâche TABSA. Nous avons présenté nos modèles basés sur l'apprentissage par transfert BERT-SentClass-M et BERT-SentClass-B. Nous avons introduit l'aspect à l'aide d'un jeton spécial dans le but de consolider toutes les informations nécessaires au modèle en une seule phrase d'entrée. Nous avons également proposé l'utilisation du modèle DistilBERT-SentClass.

Le chapitre suivant présentera les différentes performances des modèles BERT-SentClass et DistillBERT-SentClass. Une comparaison détaillée avec les modèles présentés dans la revue de la littérature sera faite.

CHAPITRE IV

PRÉSENTATION ET ÉVALUATION DES RÉSULTATS

Dans ce chapitre, nous présenterons nos résultats et nous effectuerons une comparaison des modèles proposés. D'abord, nous exposerons les résultats du modèle SVM avec les méthodes BOW et TF-IDF. Ce modèle de base sera le point de départ, permettant de mesurer l'importance du contexte, de l'ordre de mots dans la phrase d'entrée ainsi que l'équilibrage des données. Ensuite, nous évaluerons les résultats des modèles BERT-SentClass avant et après les améliorations proposées telles que le réglage fin, l'ajustement de la phrase d'entrée et la modification de la couche entièrement connectée. Enfin, nous comparerons les résultats du modèle DistilBERT-SentClass aux autres approches présentées dans le chapitre trois. Une analyse approfondie de nos résultats sera effectuée en donnant les statistiques et en analysant les échantillons erronés de l'ensemble de données Sentihood. La performance de nos modèles a été mesurée par l'accuracy stricte, la F-mesure macro-moyenne et l'AUC macro-moyenne pour la détection d'aspects tout en ignorant la classe none comme c'est convenu dans la littérature des approches ayant traité Sentihood. L'accuracy macro-moyenne et l'AUC macro-moyenne ont été mesurées pour la classification des sentiments.

4.1 Les résultats du modèle de base

Comme mentionné auparavant, le but de ce modèle est d'avoir une base de comparaison. Le tableau 4.1 présente les meilleurs résultats du modèle LR obtenus par Saeidi *et al.* (2016), ainsi que le résultat moyen de cinq exécutions des modèles SVM-A (après l'intégration de l'aspect dans la phrase), SVM-B (après avoir équilibré l'ensemble d'entraînement et gardé l'intégration de l'aspect dans la phrase) et SVM-C (après avoir équilibré l'ensemble d'entraînement et après la jonction de l'emplacement et de l'aspect) décrites dans la section 3.3.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
LR (Saeidi *et al.*, 2016)	—	**39.3**	**92.4**	**<u>87.5</u>**	**<u>90.5</u>**
SVM-A (BOW)	51.1	19.7	81.7	75.4	81.7
SVM-B (BOW)	59.3	68.9	90.1	80.8	85.1
SVM-C (BOW)	**59.8**	**69.3**	**91.4**	**81.0**	**85.8**
SVM-A (TF-IDF)	52.5	22.3	81.7	75.9	82.4
SVM-B (TF-IDF)	62.1	74.1	90.7	82.4	86.1
SVM-C (TF-IDF)	**<u>62.4</u>**	**<u>75.2</u>**	**<u>92.5</u>**	**82.4**	**86.5**

TABLEAU 4.1 – Les résultats des modèles LR et SVM.

En gras : meilleure mesure de l'approche.

Souligné : meilleure mesure pour toutes les approches.

À titre de rappel et comme expliqué dans la section 2.4.2, nous utilisons l'accuracy moyenne sur les aspects pour la classification des sentiments, l'accuracy stricte et la F-mesure macro-moyenne pour la détection d'aspects tout en ignorant la classe none. Nous évaluons la détection d'aspects comme un problème de classification multi-labels pour que les résultats soient moyennés sur les emplacements plutôt que sur les aspects.

D'après le tableau 4.1, nous remarquons que les résultats de SVM avec la méthode TF-IDF sont globalement meilleurs que les résultats avec la méthode BOW. Cette dernière crée simplement un ensemble de vecteurs contenant le nombre d'occurrences de mots, tandis que la méthode TF-IDF fournit également l'information sur les mots les plus et les moins importants. Cela nous permet de supprimer les mots qui sont moins importants sur la base d'un seuil fixe, réduisant ainsi la dimension du vecteur d'entrée.

Nous remarquons également que le modèle SVM-C est meilleur que le modèle SVM-B. Lorsque l'emplacement et l'aspect sont séparés (SVM-B), nous constatons que les deux emplacements {location1 et location2} et les quatre aspects {general, price, safety et transit-location} ne sont pas très informatifs, car ils apparaissent fréquemment dans les phrases. De plus, dans le cas où les deux emplacements sont présents dans la même phrase, le modèle SVM-B ne sera pas en mesure d'identifier l'emplacement lié à l'aspect, puisque nos deux méthodes de tokenisation ne tiennent pas compte de la position des mots dans la phrase. Ceci explique l'amélioration ap-

portée par la jonction de l'emplacement et de l'aspect du modèle SVM-C. Puisque les meilleurs résultats ont été obtenus par SVM-C avec la tokenisation TF-IDF, nous utiliserons ce modèle comme base de comparaison dans la suite de ce chapitre.

4.2 Les résultats du modèle BERT-SentClass

Dans cette section, nous présentons les résultats du modèle BERT-SentClass-M et BERT-SentClass-B avec les améliorations proprosées dans le chapitre précédent.

4.2.1 Les résultats du processus de réglage fin

Comme expliqué dans la section 3.4.2, nous avons testé nos modèles BERT-SentClass-M et BERT-SentClass-B sur l'ensemble de développement avec la méthode *early stopping* pour déterminer le nombre d'époch. Ces expérimentations ont été effectué avec différentes tailles de lot {8, 16, 24, 32} et avec différentes taux d'apprentissage en considérant un choix aléatoire uniforme dans l'échelle logarithmique. Nous avons opté pour la méthode *Random search* pour entraîner nos modèle avec des combinaisons aléatoires.

Nos expérimentations révèlent que la taille de lot n'a pas un impact considérable sur la performance des modèles BERT-SentClass-M et BERT-SentClass-B. Puisqu'il nous faut plus de temps d'exécution à chaque fois que nous augmentons la taille du lot, nous fixons la taille du lot à 8. Le choix du taux d'apprentissage, quant à lui, a un impact significatif sur les performances des modèles BERT-SentClass-M et BERT-SentClass-B. Dans la plupart des cas, une valeur de taux d'apprentissage plus faible donne de meilleurs résultats. En particulier, lorsque le taux d'apprentissage est fixé à 1e-5, nos modèles obtiennent les meilleures valeurs d'accuracy et de F-mesure sur le jeu de données Sentihood. Lorsque la valeur du taux d'apprentissage augmente, les modèles ne convergent pas et les performances diminuent. Sun *et al.* (2019a) ont également constaté qu'un taux d'apprentissage plus faible est nécessaire pour que leur modèle BERT-pair soit performant. Alors qu'un taux d'apprentissage agressif de 5e-4 ou 4e-4 ne permet pas au modèle de converger. Dans cette expérience, nous constatons la même chose, et nous arrivons aux mêmes conclusions.

Nous avons obtenu la même configuration d'hyperparamètres pour les modèles BERT-SentClass-M et BERT-SentClass-B, à savoir ; le nombre d'epochs à 7, la taille de lot à 8 et le taux d'apprentissage à 1e-5. Les résultats de nos modèles sont rapportés dans le tableau 4.2.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
BERT-pair-QA-M (Sun *et al.*, 2019a)	**79.4**	86.4	97.0	**93.6**	96.4
BERT-pair-QA-B (Sun *et al.*, 2019a)	79.2	**87.9**	**97.1**	93.3	**97.0**
BERTSentClass-M (préliminaire)	77.8	84.9	96.7	**92.8**	**96.6**
BERTSentClass-B (préliminaire)	**78.4**	**85.1**	**96.9**	92.6	96.5

TABLEAU 4.2 – Les résultats des modèles BERT-SentClass-M et BERT-SentClass-B sur le jeu de données Sentihood comparés à BERT-Pair-QA-M et BERT-Pair-QA-B de Sun *et al.* (2019a).

Avec un taux d'apprentissage fixé à 1e-5, la taille du lot à 8, la probabilité d'abandon à 0.1.
En gras : meilleure mesure de l'approche.
Souligné : meilleure mesure pour toutes les approches.

Le tableau 4.2 rapporte les deux meilleurs résultats des modèles BERT-Pair-QA-M et BERT-Pair-QA-B de Sun *et al.* (2019a), ainsi que les résultats préliminaires de nos nouveaux modèles BERT-SentClass-M et BERT-SentClass-B. Les performances de BERT-SentClass-M et BERT-SentClass-B sont légèrement inférieures à celles de BERT-Pair-QA-M et BERT-Pair-QA-B. Néanmoins, ces résultats restent largement supérieurs aux performances du modèle SVM-C de base ainsi qu'aux modèles basés sur les réseaux de neurones récurrents (biLSTM-Index et biLSTM-Final de Saeidi *et al.* (2016), Sentic-LSTM de Ma *et al.* (2018) et Dmu-Entnet de Liu *et al.* (2018)) présentés dans la revue de littérature.

4.2.2 Les résultats des ajustements de la phrase d'entrée

Le but de ces ajustements est d'améliorer les performances du modèle BERT-SentClass-M et BERT-SentClass-B précédemment exécutés avec le nombre d'epoch à 7, le taille de lot à 8, la probabilité d'abandon à 0.1 et le taux d'apprentissage à 1e-5. Le tableau 4.3 montre les résultats de nos modèles après les ajustements supplémentaires de la figure 3.6.

Les résultats obtenus dans le tableau 4.3 représentent les meilleures performances de nos modèles. Nous remarquons que l'accuracy et la F-Mesure de la détection d'aspects sont légèrement supérieures pour le modèle BERT-SentClass-B. Alors que l'accuracy et l'AUC de la détection des sentiments sont légèrement plus élevées pour le modèle BERT-SentClass-M. Ces résultats obtenus après les ajustements effectués sont supérieurs aux résultats préliminaires. Prenons l'exemple du modèle BERT-SentClass-B où l'accuracy de la détection d'aspects s'améliore de 2.1%, elle passe de 78.4% à 80.5%. La F-Mesure s'améliore de 2.2%, elle passe de 85.1% à 87.3%. L'accuracy de la classification des sentiments s'améliore de 0.5%, elle passe de 92.8% à 93.3%.

Modèles	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
BERTSentClass-M (préliminaire)	77.8	84.9	96.7	**92.8**	**96.6**
BERTSentClass-B (préliminaire)	**78.4**	**85.1**	**96.9**	92.6	96.5
BERT-SentClass-M (phrases ajustées)	80.3	86.9	**97.3**	**93.3**	**96.9**
BERT-SentClass-B (phrases ajustées)	**80.5**	**87.3**	97.1	92.9	96.6

TABLEAU 4.3 – Les résultats de nos modèles BERT-SentClass avant et après l'ajustement.

En gras : meilleure mesure de l'approche.

Souligné : meilleure mesure pour toutes les approches.

4.2.3 Les résultats de la couche entièrement connectée

Sur la base des performances des quatre modèles présentés dans le tableau 4.4, nous pouvons confirmer que l'augmentation du nombre de couches n'a aucun impact sur les résultats. D'ailleurs, la différence entre les deux modèles est faible, voire insignifiante. Pour cette raison, nous déduisons qu'il n'y a pas d'avantage à utiliser deux couches au lieu d'une et nous gardons donc le premier modèle qui est légèrement plus rapide en termes d'exécution.

Modèles	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
1 couche de classification :					
BERT-SentClass-M	80.3	86.9	**97.3**	**93.3**	**96.9**
BERT-SentClass-B	**80.5**	**87.3**	97.1	92.9	96.6
2 couches de classification :					
BERT-SentClass-M	79.9	86.9	97.1	**93.3**	96.8
BERT-SentClass-B	**80.5**	87.0	97.2	91.8	96.4

TABLEAU 4.4 – Les résultats de l'ajout de deux couches entièrement connectées.

En gras : meilleure mesure de l'approche.

Souligné : meilleure mesure pour toutes les approches.

4.2.4 Évaluation des résultats

La comparaison de la performance des modèles déjà proposés sera l'activité essentielle de cette section. En outre, une analyse approfondie de nos résultats sera effectuée tout en donnant les performances et en analysant les échantillons erronés. Pour faciliter la comparaison, nous avons regroupé tous les résultats des sections précédentes dans le tableau 4.5.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
biLSTM-Index (Saeidi *et al.*, 2016)	—	69.3	89.7	81.9	83.9
biLSTM-Final (Saeidi *et al.*, 2016)	—	68.9	89.8	82.0	85.4
Sentic-LSTM (Ma *et al.*, 2018)	67.4	78.2	—	89.3	—
EntNet (Liu *et al.*, 2018)	66.3	69.8	89.5	87.6	89.7
Dmu-Entnet (Liu *et al.*, 2018)	73.5	78.5	94.4	91.0	94.8
BERT-pair-QA-M (Sun *et al.*, 2019a)	**79.4**	86.4	**97.0**	**93.6**	96.4
BERT-pair-QA-B (Sun *et al.*, 2019a)	79.2	**87.5**	96.6	92.8	**96.9**
Average of the executions :					
BERT-pair-QA-M	78.8	86.1	96.3	92.9	**96.7**
BERT-pair-QA-B	78.9	**87.1**	96.1	92.2	96.6
BERT-SentClass-M	80.1	86.9	96.8	**93.1**	**96.7**
BERT-SentClass-B	**80.4**	87.0	**97.0**	92.4	96.5
Meilleure exécution :					
BERT-pair-QA-M	79.0	86.2	96.7	93.2	**96.9**
BERT-pair-QA-B	79.1	87.2	96.4	92.6	96.7
BERT-SentClass-M	80.3	86.9	**97.3**	**93.3**	**96.9**
BERT-SentClass-B	**80.5**	**87.3**	97.1	92.9	96.6

TABLEAU 4.5 – les résultats de la tâche TABSA sur le jeu de données Sentihood.

Le symbole "—" signifie non reporté.

En gras : meilleure mesure de l'approche.

Souligné : meilleure mesure pour toutes les approches.

D'après le tableau 4.5, nous pouvons conclure que notre nouvelle approche surpasse les modèles biLSTM-Index et biLSTM-Final de Saeidi *et al.* (2016), Sentic-LSTM de Ma *et al.* (2018) et Dmu-Entnet de Liu *et al.* (2018). D'autre part, les résultats de nos modèles BERT-SentClass-M et BERT-SentClass-B sont très similaires aux résultats des modèles BERT-pair-QA-M et BERT-pair-QA-B de Sun *et al.* (2019a), cependant, notre approche est plus simple.

Nous avons exécuté les modèles BERT-pair et BERT-SentClass cinq fois sur notre machine virtuelle Colab et nous avons rapporté la moyenne des exécutions dans la deuxième section du tableau 4.5. Nous remarquons que nos modèles BERT-SentClass-M et BERT-SentClass-B sont

légèrement supérieurs à BERT-pair-QA-M et BERT-pair-QA-B. En considérant la moyenne des exécutions, le modèle BERT-SentClass-B offre les meilleures performances pour la détection d'aspects avec 80.4% d'accuracy et 97.0% d'AUC. Le modèle BERT-SentClass-M offre les meilleures performances pour la classification des sentiments avec 93.1% d'accuracy et 96.7% d'AUC.

La troisième section du tableau 4.5 illustre les meilleures exécutions des modèles BERT-pair et BERT-SentClass sur notre machine virtuelle Colab. En comparant les meilleures performances de BERT-pair-QA-M et BERT-SentClass-M, nous constatons que le modèle BERT-pair-QA-M commet 457 erreurs de classification sur les 7516 phrases de l'ensemble de test. Alors que notre modèle BERT-SentClass-M ne réalise que 431 erreurs de classification. Nous constatons que 73% des erreurs sont communes pour les deux modèles. En nous concentrant sur les cas erronés du modèle BERT-SentClass-M, nous avons remarqué deux points : premièrement, il y a quelques erreurs d'étiquetage dans l'ensemble de test. Deuxièment, il existe des phrases ambiguës pour lesquelles il est difficile d'attribuer clairement une classe. Le tableau 4.6 expose certaines phrases élusives dans l'ensemble de test. Par exemple, les annoteurs ont attribué une étiquette *positive* à la phrase suivante : "*location1 is quite an expensive area.*". En revanche, l'étiquette correcte devrait être *negative*.

ID	Phrase	Aspect	Étiquette	Prédiction
102	**location1** have been gradually tarted up over the years and there's lots of expensive new apartments and all that.	general	*positive*	*negative*
600	theres a couple good places in **location1** but not great.	general	*none*	*negative*
710	location1 and **location2** Sorry , but it depends on the colour of your skin.	general	*none*	*negative*
914	**location1** is quite an expensive area.	price	*positive*	*negative*
1027	woohhh i live in **location1** and no i don't go around roobing/shooting people.	safety	*none*	*positive*
1489	Would not say location1 or **location2** less pleasant.	general	*positive*	*negative*

Tableau 4.6 – Extrait des erreurs de classification.

4.3 Les résultats du modèle DistilBERT-SentClass

Pendant la spécialisation de la tâche, nous avons utilisé le modèle DistilBERT-base−*Uncased* sur le jeu de données Sentihood. Ce modèle est une version distillée du modèle BERT-base−*Uncased*.

Paramètres	**BERT-Sentclass**	**DistilBERT-Sentclass**
Blocs de Transformer	12	6
taille de couches cachées	768	768
Tête d'auto-attention	12	12
Nombre de paramètres /M	110	66
Probabilité d'abandon	0.1	0.1
Nombre d'epoch	7	7
Taux d'apprentissage	1e-5	1e-5
Taille du lot	8	8

TABLEAU 4.7 – Récapitulatif des paramètres des modèles BERT-Sentclass et DistilBERT-Sentclass (Godbole *et al.* (2020) et notre précédent réglage fin).

À la lumière du tableau 4.7, nous confirmons que le nombre de blocs de Transformer du modèle DistilBERT est réduit de 50%, il passe de 12 blocs de Transformer dans le modèle BERT à 6 blocs dans le modèle DistilBERT. De plus, nous rappelons que le nombre de paramètres est réduit de 40%, il passe de 110 millions de paramètres dans le modèle BERT à 66 millions de paramètres dans le modèle DistilBERT. En ce qui concerne le reste des paramètres, c'est-à-dire la taille de couches cachées, la tête auto-attention, la probabilité d'abandon, le nombre d'epochs, le taux d'apprentissage et la taille du lot, nous gardons les mêmes valeurs pour les deux modèles. C'est grâce à la réduction des blocs de Transformer et du nombre total des paramètres que le modèle DistilBERT est exécuté plus rapidement et est considéré plus petit que le modèle BERT. Pour confirmer cela, nous avons calculé le temps d'exécution des deux modèles BERT-Sentclass et DistilBERT-Sentclass sur le jeu de données Sentihood et nous sommes passés de 3 heures d'exécution par epoch, à 75 min par epoch en utilisant le même environnement décrit dans la section 3.2. Pour faciliter la comparaison, nous avons résumé les différentes performances des modèles BERT-pair, DistilBERT-pair, BERT-SentClass et DistilBERT-SentClass dans le tableau 4.8 ci-dessous.

Modèle	Aspect			Sentiment	
	Accuracy	F-Mesure	AUC	Accuracy	AUC
BERT-pair-QA-M	79.0	86.2	96.7	93.2	**96.9**
BERT-pair-QA-B	79.1	87.2	96.4	92.6	96.7
BERT-SentClass-M	80.3	86.9	**97.3**	**93.3**	**96.9**
BERT-SentClass-B	**80.5**	**87.3**	97.1	92.9	96.6
DistilBERT-pair-QA-M	75.1	84.9	96.6	90.5	94.4
DistilBERT-pair-QA-B	74.9	**85.1**	96.6	88.9	94.3
DistilBERT-SentClass-M	75.2	85.0	96.8	**90.9**	**94.9**
DistilBERT-SentClass-B	**75.8**	85.0	**96.9**	90.7	94.7

Tableau 4.8 – Les résultats des exécutions des modèles BERT et DistilBERT.

En gras : meilleure mesure de l'approche.

Souligné : meilleure mesure pour toutes les approches.

Sur la base des résultats du tableau 4.8, il faut reconnaître que les modèles BERT et DistilBERT ont tous les deux de bonnes performances. Cependant, le modèle BERT-SentClass reste le plus performant. Nous constatons que les modèles DistilBERT-SentClass-M et DistilBERT-SentClass-B sont légèrement meilleurs que les modèles DistilBERT-pair-QA-M et DistilBERT-pair-QA-B. Par exemple, l'AUC du sentiment pour le modèle DistilBERT-SentClass-M est de 94.9%, tandis que celle du modèle DistilBERT-pair-QA-M est de 94.4%.

Nous notons que les résultats de BERT sont meilleurs que ceux de DistilBERT, il s'agit d'une différence moyenne de 2 % à 3 %. De plus, n'oublions pas qu'en utilisant le modèle DistilBERT dans notre cas, le temps d'exécution diminue d'environ 60% puisque nous passons de 3 heures d'exécution par epoch, à 75 min par epoch. Pour conclure, DistilBERT est un très bon choix si nous avons besoin de la vitesse d'exécution et si nous pouvons faire des compromis sur les mesures. Cependant, si nous recherchons les meilleures mesures, sans nous soucier du temps d'exécution ou des ressources nécessaires, BERT sera notre meilleur choix.

4.4 Conclusion

Dans ce chapitre, nous avons présenté les différentes performances des modèles SVM-A, SVM-B et SVM-C. La jointure de l'emplacement et de l'aspect du modèle SVM-C donne les meilleurs résultats avec les deux méthodes BOW et TF-IDF. Nous avons donc utilisé ce dernier comme base de comparaison. Par la suite, nous avons présenté les résultats du processus de réglage fin et de l'ajustement de la phrase d'entrée. Les résultats obtenus par les modèles BERT-SentClass-M et BERT-SentClass-B surpassent les modèles SVM-C de base, biLSTM-Index et biLSTM-Final de Saeidi *et al.* (2016), Sentic-LSTM de Ma *et al.* (2018) et Dmu-Entnet de Liu *et al.* (2018). La comparaison détaillée de BERT-pair-QA-M et BERT-SentClass-M montre que notre modèle fait moins d'erreurs de classification sur l'ensemble de test, et que 73% des erreurs sont communes aux deux modèles. Cette comparaison met également en évidence certaines erreurs d'étiquetage dans l'ensemble de test Sentihood. Finalement, nous avons présenté les performances du modèle DistillBERT-SentClass qui peut être un très bon choix si nous privilégions la vitesse d'exécution et si nous pouvons faire des compromis sur les mesures.

CONCLUSION

Dans cette étude, nous avons exploré la tâche d'analyse des sentiments à base d'aspects ciblés (TABSA). Cette tâche, comme toutes les autres tâches de traitement du langage naturel, a connu une évolution majeure depuis l'application des algorithmes d'apprentissage profond. Son rôle est d'attribuer une polarité *positive*, *négative* ou *neutre* à un aspect spécifique associé à une cible donnée.

Ce projet de recherche comporte quatre étapes principales : D'abord, nous avons présenté les approches et les techniques en apprentissage profond afin de clarifier la compréhension des concepts abordés dans notre étude. Nous avons présenté les réseaux de neurones, le concept de plongement de mots, le mécanisme d'attention, les Transformers et les mesures de performance.

Ensuite, nous avons déterminé le jeu de données Sentihood lié à la tâche TABSA et comparé les travaux réalisés par différents chercheurs. Nous avons exploré les modèles basés sur les réseaux récurrents tels que les modèles LSTM de Saeidi *et al.* (2016), Sentic-LSTM de Ma *et al.* (2018) et le modèle Dmu-Entnet de Liu *et al.* (2018). Nous avons également exploré des modèles basés sur l'apprentissage par transfert tels que BERT-pair de Sun *et al.* (2019a). Le transfert de connaissances s'est avéré efficace en termes de performance et de vitesse d'exécution par rapport aux réseaux de neurones récurrents.

Après, nous avons présenté nos différentes expérimentations sur le même jeu de données. D'ailleurs, nous avons apporté une modification à la phrase d'entrée du modèle, à savoir l'intoduction de l'aspect à l'aide d'un jeton spécial. Nous avons également effectué un réglage fin pour améliorer les performances des modèles BERT-SentClass-M et BERT-SentClass-B et nous avons proposé l'utilisation du modèle DistilBERT-SentClass.

Enfin, nous avons présenté les résultats de nos expérimentations. Notre approche consiste à insérer des balises d'une certaine manière dans la phrase d'entrée et à laisser BERT découvrir la tâche à apprendre. Les résultats de cette approche montrent que nos modèles sont stables et dépassent les résultats des architectures complexes telles que les réseaux de neurones récurrents sans l'ajout de données externes. Nos modèles BERT-Sentclass et DistilBERT-Sentclass ont réalisé des performances équivalentes aux modèles BERT-Pair de Sun *et al.* (2019a) et DistilBERT-Pair. Le

balisage des informations nous a permis de simplifier la tâche TABSA. Nous pouvons supposer que notre approche de balisage peut être appliquée dans d'autres tâches ou sur d'autres jeu de données. Nous pensons également que notre approche de balisage serait capable de fonctionner dans d'autres tâches.

En tant qu'expériences futures, il serait important d'utiliser d'autres ressources externes pour enrichir les phrases entrées et améliorer les performances de nos modèles. Il serait également intéressant, d'explorer d'autres modèles d'apprentissage par transfert comme ELMo (Peters *et al.*, 2018) et CSE (Akbik *et al.*, 2018) sur le jeu de données Sentihood. Sans compter que d'autres tests pourraient être effectués sur d'autres jeux de données.

Pour conclure, nous espérons qu'à travers ce modeste travail, nous avons pu contribuer à la recherche sur la tâche d'analyse des sentiments à base d'aspects ciblés. En effet, cette étude nous a permis d'en apprendre davantage sur le traitement du langage naturel, ce qui peut nous être d'une grande utilité dans notre vie professionnelle. Nous espérons pouvoir explorer d'autres approches du traitement du langage naturel dans un futur projet.

RÉFÉRENCES

Ahuja, R., Chug, A., Kohli, S., Gupta, S. et Ahuja, P. (2019). The impact of features extraction on the sentiment analysis. *Procedia Computer Science*, *152*, 341–348.

Akbik, A., Blythe, D. et Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. Dans *Proceedings of the 27th International Conference on Computational Linguistics*, 1638–1649., Santa Fe, New Mexico, USA. Association for Computational Linguistics. Récupéré de `https://www.aclweb.org/anthology/C18-1139`

Bahdanau, D., Cho, K. et Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Belainine, B. (2017). Classification supervisée de textes courts et bruités : application au domaine des médias sociaux.

Beltagy, I., Lo, K. et Cohan, A. (2019). Scibert : A pretrained language model for scientific text. *arXiv preprint arXiv :1903.10676*.

Cambria, E. et Hussain, A. (2015). Sentic computing. *Cognitive Computation*, *7*(2), 183–185.

Cambria, E., Poria, S., Hazarika, D. et Kwok, K. (2018). Senticnet 5 : Discovering conceptual primitives for sentiment analysis by means of context embeddings. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chung, J., Gulcehre, C., Cho, K. et Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. Dans *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186., Minneapolis, Minnesota. Association for Computational Linguistics. `http://dx.doi.org/10.18653/v1/N19-1423`. Récupéré de `https://www.aclweb.org/anthology/N19-1423`

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M. et Xu, K. (2014). Adaptive recursive neural network for target-dependent Twitter sentiment classification. Dans *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 49–54., Baltimore, Maryland. Association for Computational Linguistics. `http://dx.doi.org/10.3115/v1/P14-2009`. Récupéré de `https://www.aclweb.org/anthology/P14-2009`

Fogarty, J., Baker, R. S. et Hudson, S. E. (2005). Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. Dans *Proceedings of Graphics Interface 2005*, 129–136.

Godbole, S., Grubinska, K. et Kelnreiter, O. (2020). Economic uncertainty identification using transformers - improving current methods. Dans *seminar information Systems*. Universitat zu Berlin. Récupéré de `https://humboldt-wi.github.io/blog/research/information_systems_1920/uncertainty_identification_transformers/#comparison`

Hand, D. J. et Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, *45*(2), 171–186.

Henaff, M., Weston, J., Szlam, A., Bordes, A. et LeCun, Y. (2016). Tracking the world state with recurrent entity networks. *arXiv preprint arXiv :1612.03969*.

Hendrycks, D. et Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv :1610.02136*.

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

Kingma, D. P. et Ba, J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

Kiritchenko, S., Zhu, X., Cherry, C. et Mohammad, S. (2014). Nrc-canada-2014 : Detecting aspects and sentiment in customer reviews. Dans *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 437–442.

Kolkur, S., Dantal, G. et Mahe, R. (2015). Study of different levels for sentiment analysis.

Kris, H. et Ann, B. (2016). Sémantique distributionnelle en linguistique de corpus. *1*(201), 51–64.

Larochelle, H., Erhan, D. et Bengio, Y. (2008). Zero-data learning of new tasks. Dans *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, p. 646–651. AAAI Press.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H. et Kang, J. (2020). Biobert : a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, *36*(4), 1234–1240.

Liu, F., Cohn, T. et Baldwin, T. (2018). Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. Dans *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*, 278–283., New Orleans, Louisiana. Association for Computational Linguistics. `http://dx.doi.org/10.18653/v1/N18-2045`. Récupéré de `https://www.aclweb.org/anthology/N18-2045`

Ma, Y., Peng, H. et Cambria, E. (2018). Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Masters, D. et Luschi, C. (2018). Revisiting small batch training for deep neural networks. *CoRR*, *abs/1804.07612*. Récupéré de `http://arxiv.org/abs/1804.07612`

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D. *et al.* (2013). Ad click prediction : a view from the trenches. Dans *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1222–1230.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. et Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv :1712.09405*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. et Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv :1310.4546*.

Mikolov, T., Yih, W.-t. et Zweig, G. (2013b). Linguistic regularities in continuous space word representations. Dans *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics : Human language technologies*, 746–751.

Mohan, A. T. et Gaitonde, D. V. (2018). A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks.

Pennington, J., Socher, R. et Manning, C. (2014). GloVe : Global vectors for word representation. Dans *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543., Doha, Qatar. Association for Computational Linguistics. http://dx.doi.org/10.3115/v1/D14-1162. Récupéré de https://www.aclweb.org/anthology/D14-1162

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. et Zettlemoyer, L. (2018). Deep contextualized word representations. Dans *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237., New Orleans, Louisiana. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/N18-1202. Récupéré de https://www.aclweb.org/anthology/N18-1202

Riedl, M. et Biemann, C. (2017). There's no 'count or predict' but task-based selection for distributional models. Dans *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*. Récupéré de https://www.aclweb.org/anthology/W17-6933

Saeidi, M. (2017). Discovering and understanding community opinions of neighbourhoods expressed in question answering platforms. 176–182.

Saeidi, M., Bouchard, G., Liakata, M. et Riedel, S. (2016). Sentihood : Targeted aspect based sentiment analysis dataset for urban neighbourhoods. *arXiv preprint arXiv :1610.03771*.

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

Sokolova, M. et Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427–437.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S. et Tsujii, J. (2012). brat : a web-based tool for NLP-assisted text annotation. Dans *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 102–107., Avignon, France. Association for Computational Linguistics. Récupéré de https://www.aclweb.org/anthology/E12-2021

Sun, C., Huang, L. et Qiu, X. (2019a). Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv :1903.09588*.

Sun, C., Qiu, X., Xu, Y. et Huang, X. (2019b). How to fine-tune BERT for text classification ? *CoRR*, *abs/1905.05583*. Récupéré de `http://arxiv.org/abs/1905.05583`

Sun, S., Cheng, Y., Gan, Z. et Liu, J. (2019c). Patient knowledge distillation for bert model compression. *arXiv preprint arXiv :1908.09355*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv :1706.03762*.

Veelenturf, L. P. J. (1995). *Analysis and applications of artificial neural networks*. Prentice Hall.

Wagner, J., Arora, P., Cortes, S., Barman, U., Bogdanova, D., Foster, J. et Tounsi, L. (2014). Dcu : Aspect-based polarity classification for semeval task 4.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. et Bengio, Y. (2015). Show, attend and tell : Neural image caption generation with visual attention. Dans F. Bach et D. Blei (dir.). *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 de *Proceedings of Machine Learning Research*, 2048–2057., Lille, France. PMLR. Récupéré de `http://proceedings.mlr.press/v37/xuc15.html`