Université Paris Dauphine

Master 2 Ingénierie Statistique et Financière

Le NLP et l'étude des news financières : L'apport de l'intelligence artificielle dans la prise de décision

Pierre Fihey

Tuteur enseignant : Emannuel Lepinette

Tuteur entreprise : Jean-Baptiste Gheeraert

28 août 2023

Table des matières

1 Introduction **4**

2 Cadre de la mission d'apprentissage **5**

2.1 L'entreprise . . . 5
2.2 L'équipe Recherche Quantitative et Valorisation . . . 5
2.3 Objectif de ma mission . . . 5
2.3.1 Travail de recherche : Modification de l'achitecture d'un modèle FinBert . . . 5
2.3.2 Application : Mise en place d'un outil d'aide à la décision pour les gérants de fonds . . . 6
2.4 Remerciements . . . 6

3 Introduction au Natural Language Processing (NLP) **6**

3.1 Des premières applications aux modèles statistiques : Des débuts poussifs . . . 6
3.1.1 Des premieres avancées insuffisantes . . . 6
3.1.2 Les années 1980 et l'introduction d'une approche statistique du NLP . . . 7
3.2 Le début des réseaux de neurones et leurs applications en NLP : Le word embedding . . . 8
3.3 Le principe d'attention et l'application à de nouvelles architectures . . . 11
3.3.1 L'architecture Encoder-Decoder . . . 11
3.3.2 Le princicipe d'attention . . . 13
3.4 Introduction des modèles Transformers . . . 14
3.4.1 La self-attention . . . 14
3.4.2 L'attention multi-tête . . . 15
3.4.3 Le modèle Transformers . . . 15

4 L'architecture BERT et les modèles BART et T5, de l'analyse de langage à la génération de texte **17**

4.1 Introduction de BERT : Bidirectionnal Encoder Representations from Transformers . . . 17
4.1.1 La tokenisation . . . 17
4.1.2 L'architecture du modèle . . . 18
4.1.3 Pré-entrainement du modèle . . . 19
4.1.4 Le principe de fine-tuning . . . 20
4.2 L'Aspect-Based-Sentiment-Analysis : Usages et méthodes . . . 20
4.2.1 L'analyse de sentiments . . . 20
4.2.2 Les enjeux de la méthode ABSA . . . 21
4.2.3 Les composantes de la méthode ABSA . . . 22
4.3 BERT pour le Name-Entity-Recognition (NER) . . . 23
4.4 La génération de résumés : Le modèle BART . . . 24
4.4.1 Le modèle BART : Bidirectional Auto-Regressive Transformer . . . 24
4.4.2 La génération de résumés . . . 26
4.5 La traduction de texte : Le modèle T5 . . . 27
4.5.1 Le modèle T5 et le mécanisme de Transfer-Learning . . . 27
4.5.2 La génération de traductions . . . 28

5 FinBERT-ABSA : L'analyse de news financières **29**

5.1 Introduction . . . 29
5.1.1 Le modèle FinBERT . . . 29
5.1.2 Idée d'adaptation du modèle . . . 30

5.2 Le Dataset 32
5.3 Le modèle 32
5.3.1 NER pour la détection d'entreprise 33
5.3.2 Pré-processing des données 33
5.3.3 L'architecture du modèle 34
5.3.4 Entraînement et performance du modèle 38
5.3.5 Post-processing des données générées 40
5.4 Le pipeline final 41

6 Conclusion **43**

7 Annexes **44**

1 Introduction

Depuis le 6 décembre 2017 et la publication du papier de recherche “Attention is All You Need” [1] qui a introduit la notion d’Attention en l’incorporant dans des modèles appelés “Transformers”, le Natural Language Processing (NLP) apparaît comme l’un des sujets majeurs de recherche en Intelligence Artificielle.

En 6 ans, les progrès dans le domaine ont été considérables, stimulés par ce concept d’attention permettant de capturer les relations contextuelles entre les mots mais également par des quantités de données et de puissance de calculs toujours plus importantes.
Ces avancées techniques et matérielles ont permis aux équipes de recherche de créer des modèles constitués de plusieurs milliards de paramètres et de les entrainer sur des ensemble de données presque illimités.
L’un des principaux modèles basé sur ce principe a été introduit un an plus tard par l’équipe de recherche de Google AI Language dans le papier “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” [4]. Le modèle BERT est considéré comme une grande avancée dans le domaine du NLP puisqu’il introduit la notion d’attention bidirectionnelle permettant une meilleure compréhension du texte dans son ensemble.
Plus récemment, la compagnie OpenAI a mis en ligne l’outil ChatGPT qui témoigne des avancées spectaculaires dans ce domaine. Ce modèle, constitué de 175 milliards de paramètres et entrainé sur toutes les ressources disponibles en ligne, est capable de comprendre et de générer du texte dans plus de 100 langues avec une efficacité remarquable. Son succès ainsi que les autres modèles de language développés par les groupes Meta et Google confirment les progrès considérables qui ont été faits en 6 ans et laissent envisager de nouvelles avancées majeures qui pourraient arriver très rapidement dans le domaine du NLP.

Ces nouveaux LLM (Large Language Model) représentent une opportunité certaine en finance de marché. Les news financières sont bien évidemment scrutées chaque jour par les gérants d’actifs au sein des fonds d’investissement mais ces derniers ne peuvent analyser les news concernant chacune des compagnies dans le monde. La possibilité de créer un algorithme capable d’analyser les flux quotidiens de news dans leur totalité permettrait donc d’améliorer les prises de décision à travers la détection de tendances, de controverses ou de risques sur certaines entreprises et pourrait même permettre d’anticiper certaines fluctuations de marché.
C’est cette opportunité qui motive mon projet de recherche. En modifiant l’architecture et en ré-entrainant le modèle BERT sur des données financières, mon objectif est de mettre en place un modèle BERT appliqué à la finance et répondant à une tâche spécifique : L’Aspect-Based-Sentiment-Analysis (ABSA). Cette tâche d’analyse de sentiment demande au modèle d’extraire un sentiment spécifique à un aspect de la phrase. C’est une problématique récente dans le domaine du NLP et les modèles ABSA sont pour l’instant peu développés, surtout en finance.

Mon étude sera principalement axée sur l’analyse de controverse et de risques ESG liés à chaque entreprises. En lisant un titre de news, mon modèle doit être capable de déterminer les entreprises qui y sont mentionnées et de donner un score de sentiment sur chacune de ces entreprises.

Après avoir présenté brièvement le cadre de ma mission d’apprentissage, je vais introduire le concept de NLP en présentant les concepts majeurs qui ont permis l’avancée de la recherche dans le domaine. Je vais ensuite introduire les modèles sur lesquels se base mon travail de recherche et qui constituent l’état de l’art en NLP avant de présenter le modèle FinBERT-ABSA que j’ai pu développer pour l’analyse de news financières ainsi que le pipeline final mis en place pour la création d’un outil d’aide à la décision.

2 Cadre de la mission d'apprentissage

2.1 L'entreprise

Issue de la fusion de deux groupes de gestion d'actifs, Ofi AM et Abeille Asset Management, Ofi Invest Asset Management est une société du groupe Ofi Invest, unique pôle de gestion d'actifs du groupe Aéma.

Avec 158,1 milliards d'euros d'actifs à la fin décembre 2022, la marque Ofi Invest Asset Management constitue le 5ème groupe français de gestion d'actifs en donnant accès à une offre riche de solutions d'investissements dans le coté et le non coté et des services adaptés aux différents types d'investisseurs.

2.2 L'équipe Recherche Quantitative et Valorisation

L'équipe "Recherche Quantitative et Valorisation" a pour mission principale la valorisation des dérivés OTC et des produits stucturés, ainsi que la revue interne des modèles utilisés et l'implémentation de nouveaux modèles de valorisation.

L'équipe travaille également sur de nombreux sujets de recherche visant à utiliser des algorithmes de Machine Learning et d'Intelligence Artificielle pour des problématiques de prédiction, d'aide à la décision, de gestion des risques et d'optimisation. Cette année, l'équipe a notamment mis en place un outil de construction et d'optimisation de portefeuilles crédit IG et un outil NLP pour l'aide automatique à l'analyse de news financières.

2.3 Objectif de ma mission

Dans le cadre de mon Apprentissage, j'ai donc eu l'opportunité d'effectuer mon travail de recherche sur la production d'un nouveau modèle de NLP afin d'analyser des news financières fournies par Bloomberg avant d'incorporer ce modèle dans un pipeline composé d'une multitude de modèles de NLP. Cela dans le but de fournir aux gérants de fonds d'Ofi Invest Asset Management un outil d'analyse automatique des news financières.

2.3.1 Travail de recherche : Modification de l'achitecture d'un modèle FinBert

L'analyse de controverse et de risques ESG est aujourd'hui primordiale pour les gérants de fonds tant le marché réagit rapidement à des scandales comme des poursuites judiciaires ou des activités néfastes pour l'environnement. Or ces risques ESG ne sont aujourd'hui pas vraiment quantifiables et les données textuelles comme les articles de presses représentent donc une source importantes de données à analyser. L'objectif de mon étude est de parvenir à quantifier ces données en temps réel en extrayant de chaque article de presse les organisations qui y sont mentionnées et en leur attribuant un sentiment ESG. Les organisations les moins bien notées seront donc à éviter au sein d'un portefeuille.

Mon travail de recherche consiste donc à modifier l'architecture d'un LLM afin d'obtenir un modèle capable de résoudre cette tâche d'Aspect-Based-Sentiment-Analysis (ABSA) et de l'entrainer sur des articles de presse que j'ai labellisé au préalable. L'avantage des Large Language Model à disposition (comme BERT ou GPT) est qu'ils sont déjà pré-entrainés sur des milliards de textes. En conservant la plupart des poids pré-entrainés, cela me permet de limiter mon coût d'entrainement et d'obtenir des résultats très satisfaisant en ne labellisant que quelques milliers de titres d'articles de presse.

2.3.2 Application : Mise en place d'un outil d'aide à la décision pour les gérants de fonds

Ce travail de recherche est mené dans le but de proposer aux gérants de fonds de l'entreprise un outil d'analyse de la presse. L'objectif est que ces derniers puissent avoir une vision globale de l'actualité mais également qu'ils puissent se concentrer sur les entreprises qui constituent leurs portefeuille ou sur certains thêmes particuliers.

En ce sens, la seconde partie de mon travail consiste à incorporer mon modèle ABSA dans un pipeline composé de plusieurs autres modèles de NLP afin de fournir à l'entreprise un grand nombre d'outils pour l'analyse de news tels que la production de résumés, de traduction ou la classification thématique de l'article.

2.4 Remerciements

Je tiens à remercier chaleuresement mon maître d'apprentissage Jean-Baptiste Gheeraert ainsi que Guillaume Augé, responsable de l'equipe Recherche Quantitative et Valorisation. Jean-Baptiste et Guillaume m'ont tout de suite acceuilli avec beaucoup de bienveillance et m'ont permis d'effectuer mon travail de recherche sur un sujet qui me passione. Leurs conseils et les nombreux échanges que l'on a pu avoir sur des sujets mathématiques, financiers et d'intelligence artificelle m'ont permis de dépasser certaines difficultés mais surtout de pousser mes réflexions plus loin et de toujours faire preuve de curiosité. Cette expérience ainsi que nos discussions m'ont permis de développer réellement mon intérêt pour l'intelligence artificielle. C'est Jean-Baptiste et Guillaume qui ont motivé mon intérêt pour la recherche dans le domaine et je leur en suis réellement recconaissant.
Leur bienveillance, leur bonne humeur et leur amitié m'ont permis d'effectuer cette alternance dans les meilleurs conditions possibles.
Merci !

3 Introduction au Natural Language Processing (NLP)

Le Natural Language Processing est une branche de l'intelligence artificielle qui a pour objectif de créer des modèles capables de comprendre, analyser et générer du langage naturel. Au croisement entre la linguistique, l'informatique et la statistique, ces modèles appliquent des méthodes computationnelles à des données textuelles afin d'en comprendre le sens et de l'analyser pour répondre à de nombreuses tâches différentes comme l'analyse de sentiment, la classification thématique ou la traduction. Les avancées considérables de la recherche en Machine Learning et en Deep Learning ont permis un développement impressionant de la discipline durant les dix dernières années.

3.1 Des premières applications aux modèles statistiques : Des débuts poussifs

3.1.1 Des premieres avancées insuffisantes

Le concept de NLP a été introduit dans les années 1950. Le Mathématicien Alan Turing prédisait alors que la machine serait capable de communiquer avec l'homme de façon naturelle au 21ème siècle. Il imagina un test qui consiste à placer un observateur qui participe à une discussion avec un humain ou un ordinateur et qui doit discerner si le langage est celui d'une machine ou d'un homme. Ce test de Turing pose les objectifs que doivent se fixer les chercheurs dans la discipline.

A cette époque cependant, les chercheurs ne disposaient pas des moyens nécessaires pour envisager de tels modèles. Le premier objectif qui a motivé la recherche dans cette discipline fût de créer un outil de traduction automatique. En collaboration avec l'université de Georgetown, IBM développe en 1954 le premier algorithme capable de traduire plus de 60 phrases entre le russe et l'anglais en utilisant un dictionnaire bilingue et des règles linguistiques. Ce modèle reste encore très limité en utilisant des règles très simples mais le linguiste Américain Noam Chomsky, dans son ouvrage Syntactic Structures [15], introduit l'idée de "grammaire générative" qui énnonce des principes clairs sur lesquels pourraient s'appuyer les chercheurs en NLP. L'idée est de créer un ensemble de règles syntaxiques permettant de prédire si un texte et grammaticalement correct ou non. Ces règles peuvent être utilisées pour générer du texte et ont ouvert la voie à de nombreuses avancées dans la compréhension du langage humain.
Ce principe de grammaire générative introduit une nouvelle approche du NLP : l'approche symbolique. En Intelligence Artificielle, l'approche symbolique consiste à utiliser des règles de logique, de déduction ou de production ainsi que des symboles afin d'éxécuter des tâches.

Ces avancées ont cependant été perçues comme insuffisantes par le gouvernement américain qui publie le rapport ALPAC (Automatic Language Processing Advisory Comittee) en 1966. Ce rapport indique que les résultats ne permettent pas d'envisager la création d'un modèle de traduction automatique satisfaisant dans un futur proche et avec les moyens matériels qui existaient à cette époque. Le gouvernement décide donc de réduire drastiquement les financements alloués à la recherche en NLP.
Bien que certains modèles de langage, dans l'ensemble très basiques, aient vu le jour durant cette période, les deux décennies qui ont suivies cette réduction de financement constituent un ralentissement très fort de la recherche dans le domaine.

3.1.2 Les années 1980 et l'introduction d'une approche statistique du NLP

C'est l'essor des quantités de texte disponibles en format numérique ainsi que les progès importants en terme de puissance de calculs qui ont permis de nouvelles avancées significatives dans le domaine du NLP vers la fin des années 1980. Ces importantes bases de données ont en effet permis de mettre en place les premiers modèles statistiques appliqués au NLP, ce qui a permis de faire évoluer considérablement la recherche.

En ayant la capacité de capturer les occurences et les co-occurences de mots au sein de corpus de milliers de textes, les statisticiens ont pu mettre en place des modèles simples afin d'assigner des probabilités à des mots et des phrases. L'un des premiers modèles statistiques utilisés fût le modèle n-gram.
Le modèle n-gram est un modèle qui se base sur une simple probabilité conditionnelle. Le but est de prédire la probabilité qu'un mot soit dans la phrase en fonction des n mots qui constituent la phrase initialement. Ce modèle peut donc être utilisé afin de compléter des phrases par exemple : Soit $\mathbb{E}$ la base de mots connus par la machine et la phrase : "Le NLP est une". On suppose que l'algorithme a été pré-entrainé sur un corpus de texte, il déterminera alors le dernier mot de la phrase avec la formule suivante :

$$\text{Word} = \underset{W \in \mathbb{E}}{\text{argmax}}(\mathbb{P}[W \mid \text{Le NLP est une}])$$

. L'utilisation de ce modèle représente une avancée importante dans le domaine du NLP puisque l'algorithme est désormais capable d'apprendre par lui-même, c'est l'introduction du Machine Learning dans ce domaine. L'homme n'est plus contraint de rentrer préalablement une multitude de règles comme c'était le cas avec l'approche symbolique.

Les importantes avancées technologiques permettant des calculs plus puissants sur des bases de données plus importantes ont donc permis l'introduction des modèles statistiques dans le domaine du NLP. Même si les premiers modèles sont très basiques, comme le n-gram, les statisticiens développent également des modèles plus élaborés, c'est à cette période que l'on voit notamment apparaître les premiers HMM (Hidden Markov Model) en NLP. L'objectif est ici de prendre en compte les fonctions des mots dans la phrase. Les états cachés du modèle sont alors les mots et les observations sont les fonctions des mots. Cela permet d'obtenir une analyse plus complète d'une phrase. Le domaine du NLP prend donc une nouvelle dimension durant la fin des années 1980. Cependant, ces progrès restent limités par le fait que les bases de données ne sont constitués que de mots qui sont considérés comme des objets par la machine. Les algorithmes ne captent pas le sens des mots et ces derniers n'ont aucune valeur numérique, ce qui rend les analyses statistiques moins pertinentes.

3.2 Le début des réseaux de neurones et leurs applications en NLP : Le word embedding

L'objectif de la recherche durant la fin des années 1980 était donc de développer des solutions afin de représenter des mots ou des textes sous forme de vecteurs ou de matrices. Cela dans le but de permettre aux statisticiens de mettre en place des modèles statistiques pertinents pour l'analyse de textes mais également de permettre aux algorithmes de mieux capturer le sens des mots par exemple. Ces vecteurs représentant les mots sont appelés des Word Embeddings.

Encore une fois, les premiers modèles furent très basiques, comme la LSA (Latent semantic analysis), introduite en 1990 et qui fût l'un des premiers modèles de représentation vectorielle des mots. Le principe de la LSA est de représenter un corpus de documents sous forme de matrice où les lignes sont les mots et les colonnes sont les documents. On y trouve, pour chaque mot, le nombre d'occurence dans chaque document. L'objectif principal de la LSA est de capturer des schémas de co-occurences au sein de documents, cela peut permettre notamment de mesurer la similarité sémantique entre des mots ou des textes. Des mots possédant des vecteurs similaires ont probablement des significations proches. Si ces matrices de co-occurences représentent un outil novateur et intéressant pour les statisticiens, elles semblent tout de même trop basiques pour capturer pleinement le sens sémantique de mots ou de phrases. Une autre approche basique des words embeddings consiste simplement à hot-encoder les mots, cette méthode soulève le problème du fléau de la dimension car il faut des vecteurs de la taille du corpus de mots disponibles, ce qui peut être très coûteux et peu pertinent dans une analyse statistiques.

Il faut attendre 2003 et la publication par Bengio et Al. de l'article de recherche "A Neural Probabilistic Language Model" [10] pour voir apparaître les premiers Word Embeddings prenant en compte le sens sémantique et contextuel des mots. L'intuition de Yoshua Bengio est d'utiliser des réseaux de neurones pour définir les embeddings de mots.
L'objectif du travail de Bengio est de mesurer une probabilité conditionnelle du prochain mot d'une phrase en fonction de ceux d'avant (donc similaire à une approche n-gram vue précedemment) tout en associant à chaque mot un vecteur représentant ses différents aspects. Le but est d'optimiser les deux tâches simultanément dans un réseau de neurones basé sur l'architecture suivante :

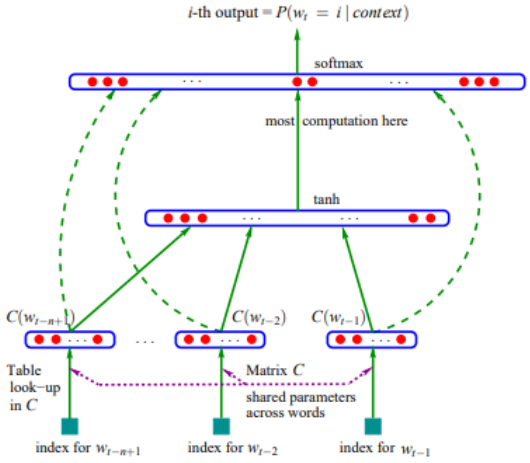


FIG. 1 – Architecture du premier réseau de neurones de langage : Bengio et Al. (2003)

La propagation forward du réseau de neurones calcule les probabilités du prochain mot en fonction des n mots d'avants qui sont fournis en input au modèle. La propagation backward a donc pour objectif de minimiser la perte en mettant à jour les poids du réseau mais également les embeddings de chaque mots. Cette mise à jour des embeddings permet d'optimiser la tâche de prédiction du prochain mot mais également de créer des embeddings représentant au mieux les relations syntaxiques et sémantiques des mots.
Cette approche de Yoshua Bengio représente une avancée majeure dans le domaine du word embeddings. D'une part, ce modèle est le premier a porposer des embeddings capturant les relations sémantiques et contextuelles des mots et améliore donc la qualité des représentation du langage. D'autre part, cela permet à la machine de créer elle même des embeddings de mots, ce qui élimine le besoin de règles manuelles ou de ressources lexicales. Surtout, ces embeddings permettent de répondre très efficacement au fléau de la dimension. En effet, la taille des embeddings est fixée au préalable et seules les valeurs du vecteur sont mises à jour, ce qui permet d'éviter le problème de dimensions que l'on pouvait rencontrer avec la méthode du hot-encoding notamment.

En 2013, l'équipe de recherche de Google dirigée par Tomas Mikolov publie le papier de recherche "Efficient Estimation of Word Representations in Vector Space" [7]. Ce papier introduit la méthode Word2Vec, une méthode qui prend ses bases sur l'approche de Bengio en 2003 mais qui a pour but de la simplifier et de l'améliorer considérablement.
L'une des premières améliorations que souhaitent apporter Mikolov et son équipe par rapport au travail de Bengio est de simplifier au maximum le réseau de neurones afin d'obtenir un réseau peu profond et entrainable sur des corpus très importants à moindre coût. Surtout, l'objectif de l'approche Word2Vec était de mettre l'accent sur la création d'embedding de mots, en prenant donc moins en compte l'approche probabiliste de Yoshua Bengio.

Le modèle présenté par Bengio [10] reposait sur une architecture de réseau de neurones feed-forward composée de la couche d'entrée, une couche de projection, une couche cachée et la couche de sortie. Les modèles feed-forwards ont la particularité de laisser circuler l'information que dans un sens, des inputs aux outputs. La complexité de calcul d'un tel modèle est très importante.
En effet, sur la couche d'entrée, les N premiers mots de la phrase (utilisés pour prédire le suivant) sont encodés, cette couche est ensuite projetée sur une couche de projection P de taille $N \times D$ avec D la dimension

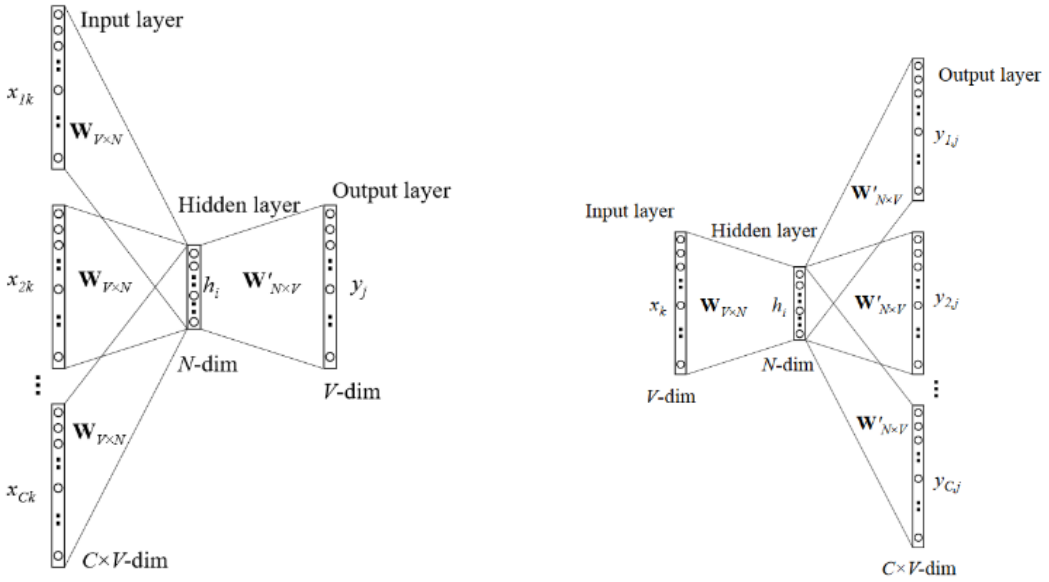
de la couche définie au préalable. En prenant N=10 (ce qui est couramment utilisé), on arrive donc à une couche de projection pouvant comporter entre 500 et 2000 neurones. La dimension H de la couche cachée étant généralement de 500 à 1000 neurones, les calculs entre ces deux couches constituent la partie la plus complexe du modèle ($N \times D \times H$). Enfin, cette couche cachée calcule la probabilité, pour chaque mot du vocabulaire, d'être le mot suivant dans la phrase, ce qui implique donc une complexité de calcul $H \times V$ avec V la taille du vocabulaire connu par la machine. Finalement, la complexité de calcul totale du modèle est : $Q_{ANN} = N \times D + N \times D \times H + H \times V$.

En 2010, Mikolov et son équipe parviennent à minimiser cette complexité en utilisant des réseaux de neurones réccurents (RNN) permettant de supprimer cette couche de projection et donc de réduire drastiquement la complexité du modèle [8]. Les RNN ont la particularité de laisser circuler l'information dans les deux sens, la couche cachée est désormais connectée à elle même, ce qui implique une complexité de calculs $H \times H$, puis à la couche de sortie. Si on considère que le modèle doit toujours donner une probabilité pour chaque mot du vocabulaire, la complexité d'un RNN pour la même tâche est donc : $Q_{RNN} = H \times H + H \times V$.

Mais le modèle Word2Vec introduit en 2013 a pour objectif de simplifier d'avantage ce modèle. La méthode adoptée pour cela est de séparer le problème en deux parties et de se concentrer, dans un premier temps, uniquement sur la création d'embeddings de mots. Afin d'arriver à une architecture optimale et la moins complexe possible, l'équipe de Google décide de supprimer la couche de projection qui rajoute une complexité importante. Tout les calculs sont donc faits sur l'unique couche cachée, ce qui permet d'obtenir une représentation des mots, certes moins précise, mais avec un entrainement beaucoup plus efficace. Pour cela, Word2Vec utilise deux architectures de modèles bien distinctes avec des tâches de prédictions qui simplifient l'apprentissage en se concentrant sur des relation locales plutôt que sur des dépendances complexe comme Bengio et Al. en 2003.

Le premier modèle utilisé consiste à prédire un mot en fonction de son contexte. Plutôt que de se concentrer sur le mot suivant dans la phrase, Mikolov décide de prendre en compte les deux cotés du mot. Le modèle prend donc N mots en entrée, tout ces mots sont encodés et la couche cachée effectue un pooling de ces vecteurs afin d'obtenir un vecteur de taille D. Un classificateur log-linéaire est ensuite appliqué à ce vecteur pour affecter à chaque mot du vocabulaire de la machine une probabilité. L'usage d'un classificateur log-linéaire plutôt qu'une fonction softmax classique est ici aussi dicté par un soucis de simplification des calculs. La complexité du modèle est donc : $Q_{CBOW} = N \times D + D \times \log_2 V$. Ce modèle est appelé CBOW (Continuous Bag of Words) car il ne prend pas en compte l'odre des mots dans le contexte.

Le deuxième modèle, appelé Continuous Skip-Gram Model, consiste lui à prendre un mot en entrée et à prédire son contexte (donc les mots qui l'entourent). Les mots les plus loins ont un poids plus faibles car une relation moins importante avec le mot en entrée. L'architecture de ce modèle est sensiblement la même que celui du modèle CBOW mais l'entrée est composée d'une unité et la sortie de N unités avec N le nombre de mots à prédire. La complexité du modèle est alors : $Q_{SG} = N \times (D + D \times \log 2V)$.



(a) CBOW **(b)** Skip-gram

FIG. 2 – Architecture des modèles utilisés dans l'approche Word2Vec

Au cours de l'entrainement de ces deux modèles, seuls les embeddings des mots sont mis à jour afin de minimiser les erreurs de prédictions. A la fin de cet entrainement, on obtient donc des embeddings de mots prenant en compte les relations sémantiques et syntaxiques entre les mots et qui capturent réeellement leurs sens. Ces réseaux ne sont donc pas des réseaux de neurones profonds mais cela permet un entrainement efficace, sur de larges bases de données et a permis d'obtenir un résultat satisfaisant pour la représentation vectorielle des mots qui est encore utilisé de nos jours. L'une des particularités notables de ce modèle Word2Vec est qu'il permet notamment de faire des opération mathématiques représentant parfaitement des opérations sémantiques. En prenant l'embedding du mot "King", en soustrayant "Man" et en ajoutant "Woman", on retrouve l'embedding du mot "Queen" par exemple.
En introduisant ces représentations vectorielles capables de capturer des relations sémantiques mais également de mettre en place des relations mathématiques entre les mots, l'équipe de Mikolov a ouvert la voie à de nombreuses avancées significatives dans la compréhension et l'analyse du langage.

3.3 Le principe d'attention et l'application à de nouvelles architectures

Le modèle Word2Vec a donc permis d'introduire une base d'embeddings de mots pré-entrainés sur un corpus très important. Ces embeddings fournis par le modèle ont très vite constitué une source importante de features pour les modèles de Machine Learning et ont permis d'accroitre fortement les performances des algorithmes. Ce nouvel outil a surtout permis de mettre en place de nouvelles architectures de modèles dans le but d'exploiter au mieux les caractéristiques des embeddings de mots et les relations qui peuvent en découler afin de répondre à des tâches complexes de NLP telles que la traduction, la classification de texte ou encore le question answering.

3.3.1 L'architecture Encoder-Decoder

C'est un an plus tard que l'équipe de recherche d'Ilya Sutskever introduit le premier modèle basé sur une architecture encodeur-décodeur dans le papier "Sequence to Sequence Learning with Neural Networks" [3]. L'objectif de ce papier de recherche est d'introduire un modèle Seq2Seq, c'est à dire un modèle capable de traiter une séquence d'entrée pour générer une autre séquence de sortie. En notant $(X_1, X_2, ..., X_T)$ la phrase

d'entrée, le but d'un tel modèle est donc de trouver la séquence $(Y_1, Y_2, ..., Y_{T'})$ qui maximise la probabilité conditionnelle :

$$\mathbb{P}[(Y_1, Y_2, ..., Y_{T'}) \mid (X_1, X_2, ..., X_T)]$$

. Ces modèles sont souvent utilisés pour les tâches de traduction ou de question answering notamment. Le modèle introduit par Ilya Sutskever a par exemple été entraîné sur une tâche de traduction de l'anglais au français.

L'intuitution sur laquelle se base ce papier de recherche est qu'un modèle Seq2Seq peut être traité efficacement avec une architecture combinant deux réseaux de neurones LSTM (Long-Short-Term Memory) bien distincts. Les LSTM sont des réseaux de neurones réccurents (RNN) conçus afin de traiter et modéliser au mieux des longues séquences de données. Ces RNN ont la particularité de posséder des cellules d'état qui permettent de conserver de l'information pendant de longues séquences. Ces cellules sont protégées par des portes qui déterminent quand l'information doit être ajoutée ou supprimée et permettent de mieux gérer les problèmes de disparition du gradient. Ce mécanisme permet au modèle de mieux capturer les dépendances à long terme qu'un RNN classique.

Le modèle introduit dans [3] est donc constitué de deux LSTM liés entre eux. Le premier LSTM est l'encodeur, son rôle est de prendre en entrée une séquence et de la traiter à travers plusieurs couches cachées. L'objectif est de capturer au mieux les informations de la séquence d'entrée au sein d'un vecteur contenu dans l'état caché du LSTM et mis à jour à chaque étape, une étape correspondant au traitement d'un mot de la séquence. Une fois que toute la séquence d'entrée a été traitée, le dernier état caché de l'encodeur est transmis au deuxième LSTM. Il s'agit du vecteur de contexte, ce vecteur doit donc contenir toute l'information nécessaire pour la prédiction et la génération de texte.
Ce vecteur est transmis au deuxième LSTM, le décodeur. Ce deuxième réseau de neurones va ensuite traiter l'information contenue dans ce vecteur afin de générer la phrase attendue en output. En notant V le vecteur de contexte fourni par l'encodeur, l'objectif du décodeur est donc de maximiser la probabilité :

$$\mathbb{P}[(Y_1, Y_2, ..., Y_{T'}) \mid V]$$

A chaque étape de décodage, le modèle prend en compte le vecteur de contexte ainsi que les éléments générés précédemment. Le deuxième mot de la phrase dépendra donc du vecteur fourni par l'encodeur mais également du premier mot qui a été généré par le décodeur. De sorte que la formule finale pour le calcul de probabilité de la séquence de sortie peut être définie par :

$$\mathbb{P}[(Y_1, Y_2, ..., Y_{T'}) \mid (X_1, X_2, ..., X_T)] = \prod_{t=1}^{T'} \mathbb{P}[(Y_t \mid V, Y_1, ..., Y_{t-1}]$$

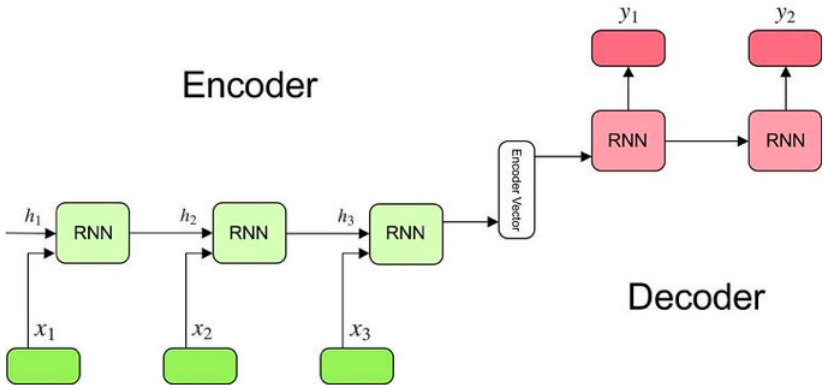


FIG. 3 – Architecture encodeur-décodeur

En permettant d'analyser des séquences de textes et d'en générer de nouvelles, cette architecture permet donc de résoudre une multitude de tâches de NLP en utilisant simplement le principe d'embeddings de mots.

3.3.2 Le principe d'attention

Si les modèles basés sur cette architecture d'encodeur-décodeur se sont avérés particulièrement efficaces dans l'étude de séquences simples, leur utilisation s'adaptait mal aux séquences plus longues ou aux groupes de phrases. En effet, l'encodeur doit réunir toute l'information de la phrase d'entrée dans un unique vecteur de contexte, ce qui peut s'avérer impossible sur des séquences de plusieurs centaines de mots. Le vecteur qui en résulte ne peut pas être vraiment pertinent pour l'analyse de la phrase et pour la génération du décodeur.

C'est en réponse à cette problématique que Dzmitry Bahdanau introduit le concept d'attention en 2014 [2]. Le principe est de fournir au décodeur des informations supplémentaires afin de spécifier, pour chaque étape de décodage, les séquences de la phrases d'entrée qui seront les plus pertinentes. Afin de mettre en place un tel modèle, Bahdanau modifie l'architecture classique des décodeurs en permettant d'adapter le vecteur de contexte à chaque étape et d'accéder à chacuns des états cachés de l'encoder et non plus juste au dernier état. On a donc, à chaque étape t, la probabilité conditionnelle :

$$\mathbb{P}[(Y_t \mid V, Y_1, ..., Y_{t-1}] = g(Y_{t-1}, S_t, C_t)$$

avec S_t l'état caché du décodeur caculé à partir des informations fournies par l'encodeur et C_t le vecteur de contexte à l'étape t.
Pour le calcul du vecteur C_t, le décodeur effectue un pooling des états cachés de l'encodeur :

$$C_t = \sum_{j=1}^{T} \alpha_{tj} h_j$$

avec h_j les états cachés de l'encodeur et α_{tj} des poids calculés selon la formule :

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})}$$

où $e_{tj} = f(S_{t-1}, h_j)$ représentent les vecteurs d'alignement. Les vecteurs d'alignements sont des vecteurs de même taille que la phrase d'entrée qui déterminent si les données d'entrée autour de la position j matchent bien avec les données de sortie autour de la position t.

A chaque étape t, le mécanisme d'attention permet donc d'extraire un vecteur de contexte qui contient les informations spécifiques à cette étape. Ce mécanisme permet d'adapter continuellement l'encodage de la phrase d'entrée et de maximiser le gain d'informations à chaque étape de génération. En introduisant ce principe, Bahdanau et son équipe permettent donc une avancée importante dans la compréhension de texte. Les chercheurs dans le domaine ont tout de suite voulu développer et améliorer ce principe et c'est en 2017 que l'équipe de recherche de Google, dirigée par Ashish Vaswani, publie le papier de recherche "Attention Is All You Need" [1] qui marque un tournant important dans la recherche en NLP en introduisant notamment l'architecture Transformers.

3.4 Introduction des modèles Transformers

En appliquant ce principe d'attention, Bahdanau a permis aux modèles Encoder-Decoder de mieux capturer l'information contenue dans chaque partie de la séquence d'entrée mais ces modèles restent limités par le fait qu'ils ne peuvent traiter les séquences que mots par mots. Cela signifie que les encodeurs et décodeurs doivent traiter les t-1 premières étapes avant de traiter l'étape t et pose une limite aux performances du modèle, surtout lorsqu'il s'agit d'étudier de larges corpus de textes.

3.4.1 La self-attention

Ashish Vaswani et son équipe ont donc décidé d'introduire un nouveau concept à ce principe d'attention afin de permettre plus de parrallélisation dans les calculs, il s'agit du concept de self-attention. La self-attention est un mécanisme spécifique qui calcule l'attention entre différents éléments d'une même séquence. L'opération de self-attention consiste en effet à attribuer à chaque mot des informations sur les mots qui lui sont le plus liés au sein même de la séquence. Le modèle introduit dans le papier [1] est donc composé de couches de self-attention qui permettent d'extraire des embeddings pertinents en appliquant les calculs suivants :

La première étape est d'obtenir trois vecteurs pour chaque vecteur fourni en input, les vecteurs Keys, Queries et Values. Pour cela, les modèles Transformers multiplient simplement les vecteurs d'input par trois matrices créees durant l'entrainement du modèle. Pour le premier mot de la phrase par exemple, en notant X_1 son embedding fourni au modèle, on obtient les vecteurs :

$$\begin{cases} \text{Query:} \quad q_1 = X_1 \times W^Q & \text{avec } W^Q \text{ la matrice Query pré-entrainée.} \\ \text{Key:} \quad k_1 = X_1 \times W^K & \text{avec } W^K \text{ la matrice Keys pré-entrainée.} \\ \text{Value:} \quad v_1 = X_1 \times W^V & \text{avec } W^V \text{ la matrice Values pré-entrainée.} \end{cases}$$

Ce sont ces trois vecteurs qui seront utilisés par le modèle pour capturer les dépendances entre les mots de la phrase. L'idée est de comparer les vecteurs keys et queries entre le mot étudié et les autres mots afin de quantifier leurs relations et d'appliquer les poids obtenus aux vecteurs values pour extraire l'information des mots qui sont en relation avec le mot étudié. En notant d_k la dimension des vecteurs keys et queries, les opérations entre ces trois vecteurs sont faites dans la couche de self-attention selon la formule suivante :

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

Cette formule d'attention constitue l'un des principaux apports de ce papier dans la recherche en NLP. Elle permet en effet de capturer les dépendances entre les mots d'une séquences de manière vraiment efficace et fait encore partie de l'état de l'art dans la matière six ans plus tard.

3.4.2 L'attention multi-tête

L'équipe d'Ashish Vaswani ne s'arrête pas à ce simple principe de self-attantion dans la mise en place de leur modèle. Ces derniers décident en effet d'améliorer la performance de la couche d'attention en effectuant ce calcul plusieurs fois pour chaque input. Pour cela, le modèle crée plusieurs sets de matrices Query/Key/Value durant l'entrainement. Chacun de ces sets possède des caractéristiques différentes et se concentre sur des relations différents entre les mots. Cela permet de maximiser le gain d'information pour le modèle en prenant en compte le plus de relations possibles. Dans le modèle introduit ici, le choix a été fait de créer huit têtes d'attention pour chaque mot.

Chaque tête d'attention est donc calculée selon la formule introduite en (1) et ces huit têtes sont ensuite concaténées afin de créer un vecteur unique d'attention. Ce vecteur est finalement multiplié par une marice de projection créee pendant l'entrainement afin d'obtenir un vecteur de bonne dimension à fournir au réseau feed-forward. Finalement, l'attention multi-têtes est donc calculée selon la formule :

$$\text{MultiHead(Q, K, V)} = \text{Concat}(head_1,...,head_h)W^O$$

avec $head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ et W^O la matrice de projection pré-entrainée.
Ce mécanisme d'attention multi-tête permet de s'assurer que les relations entre les mots sont bien capturées et cela pour un coût moindre puisque tout ces calculs sont parrallélisés.

3.4.3 Le modèle Transformers

Si ce mécanisme de self-attention multi-têtes représente déjà une avancée majeure dans la compréhension et l'analyse de texte, il ne s'agit pas du seul apport de ce papier dans le domaine du NLP. L'équipe de Google a en effet décidé de l'incorporer dans un réseau de neurones basé sur une architecture Encoder-Decoder et qui constitue encore aujourd'hui l'un des modèles les plus utilisés dans le domaine.

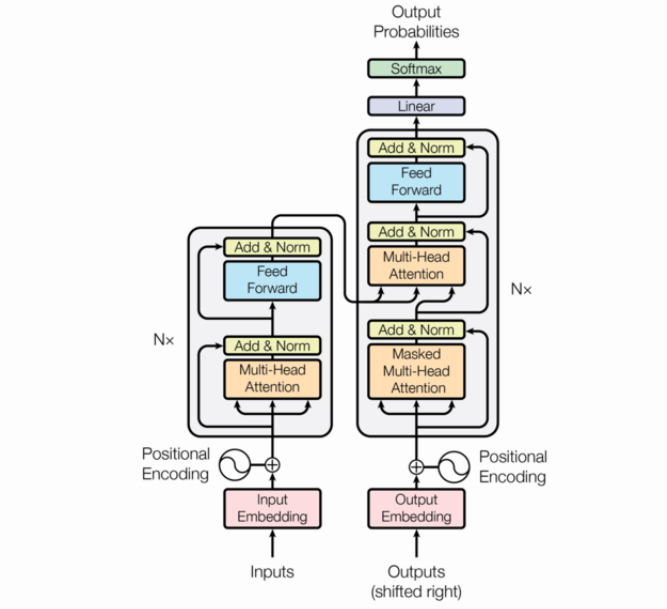


FIG. 4 – Architecture du modèle Transformers

Ce modèle est donc constitué d'un encodeur de 6 couches contenant une couche d'attention multi-têtes

introduite précédemment et une couche Feed-Forward classique. Ces couches d'attention et feed forward sont toujours suivies d'une couche "Add and Norm" qui applique la fonction suivante :

$$\text{Output}_{AddNorm} = \text{LayerNorm}(X + \text{Sublayer}(X))$$

avec Sublayer(X) l'output de la couche précédente.

Cet encodeur fournit finalement au décodeur un embedding de 512 valeurs pour chaque mot de la séquence d'entrée. C'est l'une des principales différences avec les modèles précédents dans la mesure où il n'y a plus simplement qu'un vecteur de contexte mais un vecteur pour chaque mot.

L'ensemble de ces embeddings est ensuite fourni au décodeur. Ce dernier est également constitué de 6 couches mais celles-ci contiennent une couche supplémentaire par rapport à l'encodeur. Le décodeur prend en entrée les embeddings fournis par l'encodeur mais également, à chaque étape, les embeddings des étapes précédentes. A l'étape t, la couche "Masked Multi-Head Attention" applique donc le mécanisme de self-Attention sur les embeddings des t-1 premiers mots générés. L'output de cette couche est ensuite envoyé vers la couche d'attention multi-têtes classique qui effectue l'opération (1) sur cet output ainsi que sur les embeddings fournis par l'encodeur. Après normalisation, les embeddings sont fournis à une couche feed-forward classique. Finalement, les embeddings de la dernière couche du Decodeur sont envoyés vers une couche linéaire qui les projette sur un vecteur de la taille du vocabulaire connu. La fonction Softmax est appliquée à ce vecteur afin d'obetnir des probabilités pour chaque mot et de générer le mot qui maximise ces probabilités.

On remarque également qu'avant d'incorporer les embeddings de mots à l'encodeur ou au décodeur, le modèle leur applique un encodage positionnel. Cet encodage permet au modèle de prendre en compte la distance entre les mots dans un texte. Deux mots éloignés sont en effet moins susceptibles d'être très dépendants en général. Pour faire cela, le modèle additionne simplement un vecteur positionnel à chaque embedding de mot.

En introduisant ces mécanismes de self-attention, d'attention multi-têtes et en les incorporant dans un réseau de neurones permettant un échange constant entre l'encodeur et le décodeur, l'équipe de recherche de Google a donc permis de mettre en place un modèle capable de capturer efficacement les dépendances complexes entre les mots d'une séquence, ceci même au sein de larges textes. Ce modèle Transformers permet également un calcul parrallèle qui rend l'entrainement nettement plus efficace. La publication du papier "Attention is all you need" est donc perçue comme un tournant majeur dans la recherche en NLP et a permis des améliorations spectaculaires dans les performances des modèles de traitement de langage. Toutefois, ces modèles Transformers ont été entrainés sur des données labellisées et sont particulièrement efficaces pour des tâches "sequence-to-sequence". Les modèles sont entrainés pour générer une séquence à partir d'une autre, étape par étape. C'est afin de répondre à des tâches différentes, plus spécifiques à l'analyse de texte, que le papier "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [4] a été publié en 2018 par une autre équipe de recherche de Google dirigée par Jacob Devlin.

Avec l'apparition des word embeddings qui ont permis l'usage de réseaux de neurones dans le traitement de langage mais également l'introduction d'architectures de modèles permettant de capturer le sens des mots et les relations contextuelles au sein des phrases, la recherche en NLP a connu un réel essor dans les années 2010. L'introduction des Transformers constitue une étape majeure dans la mesure où ce modèle

ouvre la voie à de nombreux modèles d'analyse de langage se basant sur ces principes d'attention et sur cette architecture pour atteindre des performances vraiment satisfaisantes. C'est notamment le cas du modèle BERT que je vais présenter en prochaine partie, c'est sur ce modèle que se base mon travail de recherche.

4 L'architecture BERT et les modèles BART et T5, de l'analyse de langage à la génération de texte

Comme énnoncé en introduction, l'enjeu de mon travail de recherche est de créer un modèle capable d'extraire un sentiment ESG spécifique à chaque entreprise mentionnée dans un texte. Ce travail de recherche sera ensuite introduit dans un pipeline de modèles permettant d'offrir un outil pertinent d'analyse de presse aux gérants de fonds de l'entreprise. Afin d'effectuer cette analyse, j'ai décidé de me baser sur le modèle BERT. Ce dernier offre en effet des capacités importantes de compréhension de langage et et des possibilités d'adaptation pour traîter toutes sortes de tâches d'analyse de langage. Pour les problématiques de génération de langage comme la traduction ou la production de résumés, le modèle BERT n'est cependant pas adéquat et je vais donc introduire deux autres modèles qui apportent des modifications vraiment pertinentes aux modèles Transformers, les modèles BART et T5.

4.1 Introduction de BERT : Bidirectionnal Encoder Representations from Transformers

Le modèle BERT est un modèle de représentation de langage, c'est à dire que l'objectif de ce modèle est simplement de prendre des données textuelles en entrée et d'en extraire des vecteurs denses contenant les caractéristiques sémantiques et contextuelles du texte afin de l'analyser selon la tâche définie. Cet objectif diffère de celui des premiers Transformers qui sont basés sur une architecture Encoder-Decoder afin de pouvoir générer des phrases à partir des phrases d'entrée.

4.1.1 La tokenisation

Avant d'étudier plus en détail les mécanismes du modèle BERT, il semble important d'introduire le concept de tokens. Le mécanisme de tokenisation désigne le fait de séparer une phrase en plusieurs morceaux qui seront fournis au modèle pour l'analyse. Les modèles tels que les Transformers ne prennent en effet pas en entrée une seule chaîne de caractère contenant tout le texte mais plusieurs chaînes distinctes qu'ils analysent parrallélement. Surtout, ces tokens permettent de faire le lien entre le texte et les représentations vectorielles nécessaires pour l'utilisation d'un réseau de neurones. Chaque token d'un vocabulaire est en effet associé à un embedding pré-entrainé qui contient le sens de ce mot. Le modèle attribue à chaque token son vecteur associé afin d'entrer dans la première couche du réseau de neurones.

La séparation de textes en tokens n'est pas unique et de nombreuses méthodes peuvent être utilisées selon le contexte envisagé.
La méthode la plus basique consiste à séparer le texte caractère par caractère. Cette méthode représente un avantage principal, la base de vocabulaire ne doit contenir que les caractères de la langue étudiée (quelques dizaines de tokens contre des dizaines de milliers pour une base de mots) et tout les mots peuvent être construits à partir de ces caractères. Cependant, cette méthode ne peut pas être efficace dans la mesure où chaque caractère ne signifie rien individuellement, le modèle pourra donc extraire une relation entre les caractères mais il est presque impossible de capturer un sens sémantique en analysant une phrase lettre par lettre. De plus, cette méthode implique que les inputs fournis au modèles seront de trop grandes tailles et exigeront des calculs plus importants.

Une autre approche très intuitive et qui représente l'une des approches les plus utilisées en NLP consiste à séparer le texte mot par mot. Cette méthode semble plus intéressante dans la mesure où chaque mot du vocabulaire est associé à un vecteur contenant les informations sur ce mot. En séparant le texte mot par mot, il devient donc facile pour le modèle d'étudier les relations contextuelles entre les mots et de capturer pleinement le sens de la phrase. Cependant, cette approche présente un inconvénient majeur. Si l'on souhaite associer chaque mot d'un corpus de texte à un token d'un vocabulaire, il faudrait disposer d'une base de données de plusieurs millions de tokens. Ce problème se présente notamment lorsqu'il faut tokeniser un verbe par exemple. Une telle méthode nécessite en effet que chacune des formes conjuguées du verbe soit présente dans le vocabulaire.

Ces deux méthodes constituent donc des approches pertinentes de la tokenisation mais présentent des inconvénients majeurs pour les modèles d'analyse de langage. Pour tirer le meilleur de ces deux approches et limiter les inconvénients, une troisème méthode a été mise en place, la "subword tokenization". Cette méthode consiste à séparer le texte mot par mot mais également à séparer certains mots de manière logique. Le mot "Lower" sera par exemple séparé en "Low-" et "-er", le modèle est alors en capacité de comprendre que le suffixe représente un outil de comparaison. Il en est de même pour les verbes conjugués qui sont séparés entre la racine et la terminaison. Cela permet au modèle de capturer parfaitement le sens des mots, tout en disposant d'une base de vocabulaire limitée puisqu'il ne doit connaître que les racines des mots, ce qui réduit considérablement le nombre de tokens différents. C'est cette approche de "subword tokenisation" qui est utilisée dans la plupart des modèles Transformers. Le modèle BERT utilise une variante de cette approche, introduite sous le nom de "WordPiece" dans le papier de recherche "Japanese and Korean Voice Search" [6] en 2012.

4.1.2 L'architecture du modèle

Si les modèles Transformers et le modèle BERT ne partagent pas complètement les mêmes objectifs, l'architecture et les mécanismes d'attention introduits dans "Attention is All You Need" [1] représentent toujours l'état de l'art dans la compréhension de texte et la capture de relations et de dépendances complexes entre les mots. L'équipe de Jacob Devlin a donc décidé de conserver ces mécanismes mais en utilisant uniquement l'encodeur. Cet encodeur fournit en effet des embeddings contenant toute l'information nécessaire sur le texte d'entrée pour traîter de nombreuses tâches de NLP comme la classification de texte ou l'analyse de sentiments.
Le modèle BERT est donc constitué des mêmes couches que l'encodeur du modèle Transformers classique, seuls les hyperparamètres différent quelques peu. Tout d'abord, le modèle BERT est constitué de 12 couches Transformers (contre 6 pour l'encodeur Transformers). Chacune des couches est constituée d'une couche "Multi-Head Attention" et d'une couche Feed-Forward. Cependant, chacune de ces couches fournit un embedding de 768 valeurs (contre 512 pour l'encodeur Transformers) et les couches d'attention comportent 12 têtes d'attention (contre 8).

Mais la principale avancée que représente le modèle BERT se trouve dans l'entrainement du modèle, ou plutôt dans le pré-entrainement. L'objectif est en effet de créer un modèle pré-entrainé sur une base de données presque illimitée afin de permettre de capturer au mieux le sens de mots et leurs dépendances tout en laissant la possibilité de ré-entrainer ces poids sur des tâches ou des textes spécifiques afin d'adapter au mieux cette compréhension du texte à chaque problème, c'est le principe de finetuning.

4.1.3 Pré-entrainement du modèle

Afin de mettre en place ce modèle bidirectionnel capable de prendre en compte les mots suivants pour l'étude d'un mot précis, BERT a été entraîné sur deux tâches bien précises ne nécessitant aucune labellisation du jeu de données et permettant donc un entrainement sur un très vaste corpus (3,3 Milliards de mots).

La première tâche d'entrainement consiste à masquer aléatoirement un mot dans une phrase et d'entraîner le modèle à prédire ce mot. On appelle cette procédure un Masked Language Model (MLM). C'est cette tâche qui introduit le principe de bidirectionnalité dans le modèle BERT. En effet, les mots masqués ne sont pas forcèment les derniers et le modèle dispose donc de tout les mots de la phrase pour le prédire. Une telle approche avait été utilisée dans le modèle Word2Vec [7] mais ne consistait qu'à prendre tout les mots de la phrase dans un "Bag-of-words", c'est à dire que le modèle ne prenait pas en compte les positions des mots. Le modèle BERT utilise les encodages de positions introduits par les modèles Transformers et permet donc de prendre en compte le fait que les mots viennent après dans la phrase, c'est une amélioration significative par rapport au modèle Word2Vec de Mikolov.
L'entrainement de BERT à cette tâche MLM consiste à masquer aléatoirement 15% des mots du corpus d'entrainement. Sur l'ensemble de ces mots masqués, 80% sont remplacés par le token [MASK], 10% sont remplacés par un mot aléatoire et 10% restent inchangés. Cette nuance dans le masque de mots permet de renforcer la robustesse du modèle en l'entrainant à discerner les contextes cohérents et moins cohérents. Cela force également le modèle à étudier toute les mots de l'input (et non pas seulement les mots remplacés par [MASK]) afin de déterminer si les mots sont cohérents ou non.
Pour effectuer cet entrainement, une simple couche de classification (Fully connected layer) est appliquée aux embeddings fournis par l'encodeur BERT, une fonction softmax est ensuite appliquée à cette couche pour extraire des probabilités pour chaque token et générer celui qui maximise cette probabilité.

La deuxième tâche sur laquelle le modèle BERT est pré-entrainé consiste à fournir au modèle deux phrases, ce dernier doit prédire si la deuxième phrase suit directement la première dans un texte. C'est la tâche de Next Sentence Prediction (NSP). Cette tâche permet d'apprendre au modèle à comprendre les relations sémantiques et contextuelles entre les phrases au sein d'un même texte. Le modèle apprend donc à capturer des relations entre les phrases mais également à comprendre les enchainements logiques au sein d'un texte.
Afin d'entrainer le modèle BERT sur cette tâche, l'équipe de Jacob Devlin a simplement constitué une base de données constituée de paires de phrases, 50% des paires sont correctes et 50% sont fausses.

Le modèle BERT a donc été pré-entrainé en résolvant ces deux tâches sur un corpus total de 3,3 milliards de mots. Cet entraînement a permis de comprendre le sens et le contexte bidirectionnel des mots, de capturer des relations complexes entre les mots mais également entre les phrases d'un texte. L'équipe de recherche de Google a donc obtenu un modèle qui constitue l'état de l'art dans le domaine de la compréhension et de l'analyse de texte. De plus, cet entraînement sur une telle base de données est très coûteux et impossible à réaliser sans les moyens dont disposent Google. Ces derniers ont en effet entraîné le modèle sur 64 TPUs (Tensor Processing Units) durant 4 jours. En effectuant ce pré-entrainement, l'équipe de Devlin a donc délivré un modèle dans lequel les poids des couches de neurones contiennent des informations pertinentes sur la langue et les relations entre les mots. Ces poids ont la possibilité d'être conservés ou adaptés à de nombreuses tâches spécifiques en utilisant très peu de données et de coût de calcul, c'est le principe de fine-tuning.

4.1.4 Le principe de fine-tuning

Le fine-tuning d'un modèle est une méthode qui consiste à ajuster les paramètres d'un modèle, parfois en rajoutant une couche supplémentaire afin de résoudre des tâches spécifiques. Le coût d'une telle opération est moindre par rapport à un coût d'entrainement global d'un modèle. Ce paradigme pré-entraînement/finetuning constitue l'un des principaux apports du papier publié par Jacob Devlin et Al. [4]. Le modèle BERT offre en effet la possibilité d'être adapté pour une grande diversité de tâches de NLP telles que l'analyse de sentiments, la prédiction de textes, la classification de textes, le résumé de texte et a vite obtenu de meilleurs performances que les modèles existant dans chacune de ces tâches.

Le fine-tuning du modèle ne nécessite que très peu de calculs et de moyens. La première étape est de créer la couche de sortie adaptée au problème à résoudre (une couche de classification pour un problème d'analyse de sentiment par exemple ou une couche softmax pour un problème de génération de texte). Une fois cette couche crée, le modèle peut être ré-entrainé sur un ensemble de données de quelques milliers de phrases labellisées et en quelques epochs seulement. Pour cela, le modèle offre la possibilité de geler certains poids, c'est à dire de demander explicitement au réseau de neurones de ne pas mettre à jour ces poids lors de l'entrainement. Si la comprehension globale proposée par BERT est satisfaisante et qu'il ne suffit que d'entrainer la classification finale, les douze couches d'encodeurs peuvent donc être gelées et seuls les poids de la dernière couche seront mis à jour.
Il est également possible de ne pas geler les poids afin de permettre au modèle de les modifier légèrement afin d'adapter la compréhension du langage dans un certain contexte. Dans un problème de classification dans le domaine de la finance par exemple, il peut être utile de permettre au modèle de s'adapter légèrement à un langage et un contexte financier.

En instoduisant le principe de bidirectionnalité dans l'analyse de phrase et ce paradigme pré-entrainement/finetuning, Jacob Devlin et son équipe ont développé un outil permettant à la communauté de chercheurs en NLP mais également aux professionnels de tout secteurs de créer des modèles efficaces et capables de répondre à de très nombreuses tâches, même avec des moyens limités. Ce papier de recherche constitue donc un tournant dans le domaine du NLP et va permettre de créer de nombreux modèles, c'est l'ambition de mon travail qui va s'appuyer sur le modèle introduit par Jacob Devlin.

4.2 L'Aspect-Based-Sentiment-Analysis : Usages et méthodes

Mon travail de recherche consiste à résoudre une tâche d'analyse de sentiment spécifique à un aspect de la phrase, je veux capturer l'information de la phrase par rapport à l'entreprise qui y est mentionnée. Une telle tâche est définie dans la littérature sous le nom d'Aspect-Based-Sentiment-Analysis (ABSA). Bien que l'analyse de sentiments classique constitue l'une des problématiques majeures dans en NLP, les modèles ABSA sont moins nombreux, surtout dans le domaine de la finance.

4.2.1 L'analyse de sentiments

Avant d'étudier le concept d'ABSA, il convient d'introduire la tâche d'analyse de sentiments, l'une des principales tâches traitées dans le domaine du NLP.
Le Sentiment Analysis (ou Opinion Mining) consiste simplement à attribuer un score, souvent Positif/Neutre/Négatif à un texte afin d'en extraire des informations sur le sentiment général. Les applications possibles pour les modèles entrainés sur cette tâche sont importantes. Les entreprises peuvent en tirer profit pour analyser les avis des clients sur leurs produits par exemple, même les partis politiques peuvent utliser de tels

modèles afin d'étudier les flux de messages sur les réseaux sociaux notamment.

L'une des premières méthodes implémentées pour une telle tâche consiste à affecter à chaque mot du vocabulaire un score de sentiment et d'analyser les phrases comme cela. Cette idée a été développée par Finn Arup Nielsen en 2011. Ce dernier a décidé d'affecter à chaque mot d'une base de vocabulaire un score entre -5 et 5. La base comporte environ 3300 mots. C'est la méthode d'analyse de sentiment la plus simple possible puisqu'il suffit d'agréger les scores de chaque mot de la phrase pour obtenir un sentiment global. Cette méthode n'est cependant pas très pertinente et l'apparition de modèles tels que les Transformers ou le modèle BERT ont permis de créer des modèles bien plus performants dans la résolution de cette tâche.

BERT constitue en effet un modèle vraiment pertinent pour l'analyse de sentiment. Comme énnoncé précédemment, ce modèle est pré-entrainé pour fournir des embeddings contenant les informations contextuelles et sémantiques de chaque mot de la phrase. En générant de tels vecteurs, le réseau de neurones rend la tâche de classification facile puisque assimilable à un problème de Machine Learning classique. En ajoutant une simple couche de classification au modèle BERT, il est donc possible d'obtenir un modèle très performant dans la classification de sentiments. Les caractéristiques de ce modèle permettent surtout d'entrainer cette classification sur un jeu de données réduit ne contenant que quelques milliers de phrases afin d'adapter les poids au mieux. Cela peut s'avérer vraiment utile dans la mesure où une classification de sentiments peut s'avérer dufférente selon les points de vues envisagés. Un modèle d'analyse de sentiment classique ne pourra par exemple pas prendre en compte la problématique ESG importante dans mon étude. En fournissant la phrase "Apple invests billions in renewable energies" au modèle de classification classique basé sur le modèle BERT, on obtient par exemple le résultat suivant :

FIG. 5 – Sentiment Analysis : BERT model (1)

Logiquement, le modèle classique ne voit pas un investissement en énergies renouvelables comme un point forcèment positif et retourne donc le sentiment "neutre". Mais en finetunant mon modèle sur quelques milliers de phrases labellisées pour prendre en compte ce critère, les poids seront légèrement adaptés pour fournir le résultat attendu. Cette capacité d'adaptation de ce modèle en fait donc un candidat parfait pour une tâche comme l'analyse de sentiment qui peut souvent s'avérer subjective.

4.2.2 Les enjeux de la méthode ABSA

La tâche d'Aspect-Based-Sentiment-Analysis représente un réel enjeu dans plusieurs secteurs. En permettant aux modèles d'étudier les sentiments spécifiques à un aspect d'un texte, cette méthode peut permettre aux entreprises ou aux restaurateurs d'analyser plus efficacement les avis laissés par les clients par exemple. Si un client laisse un avis sur un ordinateur en indiquant que la batterie est parfaite mais que l'écran pose problème, un modèle ABSA peut permettre à l'entreprise d'analyser correctement cet avis en attribuant un score pour chaque aspect (batterie et écran). C'est d'ailleurs pour de telles applications que la plupart des

modèles ABSA sont développés, les modèles sont généralement entraînés et testés sur le dataset SemEval constitué d'avis laissés par des clients sur des ordinateurs et des restaurants.

Si peu de papiers documentent l'utilisation de ce mécanisme dans le domaine de la finance (il n'existe d'ailleurs aucun dataset dans ce domaine pour entrainer des modèles ABSA), son utilisation peut s'avérer vraiment utile pour l'étude de news financières. Dans l'optique d'analyser les titres de ces news afin d'obtenir des sentiments pour chaque entreprise, l'utilisation de tels modèles est effectivement nécessaire et peut s'avérer très efficace. Si les phrases ne comportant qu'une seule entité peuvent être étudiées à l'aide d'un simple algorithme d'analyse de sentiments (même si cela sera moins précis), le même travail est impossible pour une phrase comportant deux entités. En utilisant le modèle de classification classique basé sur le modèle BERT, on s'aperçoit par exemple que ce dernier ne peut être utilisé dans le cadre de notre étude :

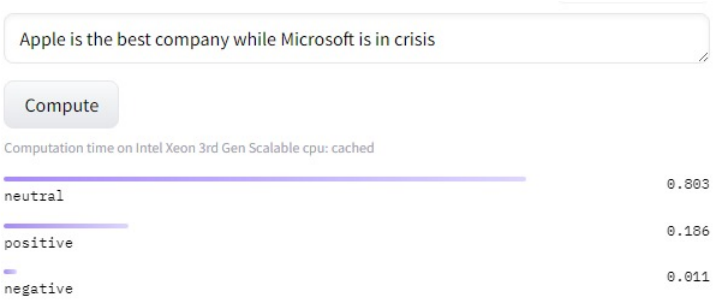


FIG. 6 – Sentiment Analysis : BERT model (2)

La phrase "Apple is the best company while Microsoft is in crisis" contient deux entités : Apple et Microsoft. Le modèle BERT classique n'est pas capable de capturer la nuance de sentiments spécifique à chacune des entreprises, il a été entrainé à extraire un sentiment pour la phrase dans sa globalité. Pour une problématique telle que la mienne, il est donc nécessaire de mettre en place un modèle capable de résoudre cette tâche d'Aspect-Based-Sentiment-Analysis afin d'obtenir un sentiment spécifique à chaque entreprise.

4.2.3 Les composantes de la méthode ABSA

Dans un papier destiné à répertorier et expliquer les méthodes ABSA [9], Wenxuan Zhang et son équipe introduisent les quatres composantes qui peuvent être prises en compte dans la création d'un modèle pour cette tâche. Le but d'un modèle ABSA étant de predire un sentiment pour un aspect de la phrase, les deux principales composantes sont évidemment l'aspect et le sentiment. Mais selon Wenxuan Zhang, chacune de ces composantes peut être séparée en deux éléments différents.
L'aspect peut en effet être étudié de deux manières : "L'aspect category" qui consiste à attribuer une catégorie au mot étudié et "l'aspect term" qui consiste simplement à extraire le mot que l'on veut analyser. Dans la phrase "La pizza est délicieuse", l'aspect term serait donc "pizza" tandis que l'aspect category serait "food" par exemple.
De la même façon, le sentiment peut également être étudié de deux manières : Le "sentiment polarity" qui consiste à attribuer un score "positif", "neutre" ou "négatif" à l'aspect et "l'opinion term" qui consiste à extraire simplement le terme de la phrase qui caractérise le mieux l'aspect. Dans la même phrase que précedemment, le sentiment polarity serait donc "positif" tandis que l'opinion term serait "délicieuse".

Dès lors, il existe différentes tâches ABSA à résoudre. L'auteur de ce papier sépare ces tâches en deux ensembles : Les tâches ABSA "simples" et les tâches ABSA "composées".
Les tâches simples consistent à rechercher uniquement l'une des quatres composantes citées précédemment.

Le modèle devra simplement extraire un terme (ou catégorie) ou un sentiment(ou terme d'opinion) par rapport à ce terme.
Les tâches composées sont plus complexes, le modèle doit extraire à la fois les termes (ou catégorie) et les sentiments (ou terme d'opinion) qui leurs sont liés.

Mon étude consiste donc en une tâche ABSA composée. En prenant une phrase en entrée, le modèle doit pouvoir extraire un terme (une entreprise) et lui attribuer un sentiment (positif/neutre/négatif). Le papier publié par Wenxuan Zhang attribue à cette tâche le nom de "End-to-End ABSA Task" (E2E-ABSA). J'ai cependant décidé de diviser cette tâche en deux modèles distincts que j'ai mis en commun dans un pipeline afin de résoudre cette tâche.

4.3 BERT pour le Name-Entity-Recognition (NER)

Comme indiqué précédemment, l'un des intérêts majeurs du modèle BERT réside dans sa capacité à s'adapter afin de résoudre de nombreuses tâches d'analyse de langage bien distinctes. En permettant de finetuner un modèle pré-entrainé sur un tel corpus afin de l'adapter à la tâche souhaitée, Jacob Devlin et son équipe ont délivré un outil vraiment efficace pour l'analyse de texte dans son ensemble.
C'est donc en m'appuyant sur des modèles BERT réentrainés sur des tâches précises que j'ai pu traîter la tâche d'E2E-ABSA (De la reconnaissance d'entreprise à l'analyse de sentiments).

La tâche de Name-Entity-Recognition consiste à extraire d'une phrase ou d'un texte les entités qui y sont mentionnées et de les attribuer à une catégorie prédéfinie. Ces catégories sont en général plutôt simples et classiques, les modèles sont entrainés à extraire les noms de lieux, d'entreprises ou les noms propres par exemple. Cependant, les modèles permettant des réentrainements à moindre coût comme BERT permettent d'adapter cette tâche à de nombreux domaines. En médecine par exemple, des modèles ont été réentrainés afin d'extraire les termes médicaux ou les noms des médicaments d'un rapport par exemple.

Encore une fois, les méthodes de résolution d'une telle tâche sont nombreuses et ont évolué depuis le début de la recherche sur le sujet.
L'une des premières approches consistait à se baser sur un dictionnaire de mots labellisés, si le modèle extrait l'un de ces mots de la phrase, il peut alors facilement lui attribuer une catégorie en se référant au dictionnaire. Cette approche est basique et efficace mais nécessite une quantités de données labellisées très importante et à mettre à jour constamment ce qui constitue une limite importante.
Une autre méthode, plutôt basique également, consiste à mettre en place des règles logiques, soit sur la forme des mots eux mêmes (un mot avec une majuscule sera attribué à la catégorie nom propre par exemple) soit sur le contexte des mots (si un mot est précédé du mot "Mr", on peut s'attendre à ce qu'il s'agisse du nom de la personne).

Mais ces deux méthodes présentent de nombreuses limites notamment par le fait qu'elles considèrent les mots comme des objets et ne permettent pas de tirer profit des word embeddings par exemple. L'apparition des modèles Transformers et BERT ont à nouveau permis d'obtenir des performances très intéressantes pour la résolution de cette tâche.
Encore une fois, la possibilité de fine-tuner le modèles BERT a joué un rôle important dans ces performances en limitant le nombre de données à labelliser pour le ré-entrainement. Une équipe de recherche de l'université de Hong-Kong a cependant décidé de pré-entrainer un modèle BERT sur une base de données regroupant toutes les bases en open-source afin de maximiser les performances du modèle et de le rendre adaptable

à tout les secteurs (Biomédical, politique...) pour une tâche NER [11]. Zihan Liu et son équipe ont tout de même tiré profit des capacités de BERT puisqu'ils ont initialisé le modèle selon les poids pré-entrainés, ils ont simplement décidé de mettre à jour tout les poids durant l'entrainement plutôt que d'entrainer simplement la couche de classification.
Un tel modèle est entrainé sur des données labellisées de manière un peu particulière, c'est le mécanisme d'annotation IOB (Inside-Outside-Beginning). Ce mécanisme consiste à attribuer à chaque token sa place dans le mot ou l'expression dont il est issu. Cela permet au modèle de capturer le début et la fin de chaque entité dans une séquence de tokens et facilite donc la classification de ces entités. Pour la phrase, "Microsoft Corporation is located in New Mexico", le dataset d'entrainement serait donc labellisé comme cela :

Microsoft	Corporation	is	located	in	New	Mexico
B-Org	I-Org	O	O	O	B-Loc	I-Loc

Avec B : Beginning, I : Inside, Org : Organisation, Loc : Localisation.

Une fois le dataset labellisé de la sorte, Zihan Liu et son équipe ont simplement rajouté une couche finale de classification d'entités sur le modèle BERT avec autant de neurones que de catégories souhaitées. Disposant d'assez de données et de puissance de calculs pour ré-entrainé complètement le modèle, ils ont décidé de mettre à jour tout les poids de BERT. En faisant cela, cette équipe de recherche a maximisé les performances du modèle sur la tâche de NER. En finetunant uniquement la dernière couche sur des données de secteurs spécifiques, ce modèle a finalement permis d'obtenir des meilleures performances que les modèles BERT classiques. C'est un modèle de la sorte que je vais intégrer dans mon pipeline pour la détection d'entreprises dans les titres de presse.

Si l'architecture du modèle BERT représente un intérêt certain pour des tâches de traitement de langage telles que l'analyse de sentiment ou le NER, ce modèle n'est cependant pas constitué de décodeur, ce qui rend impossible la génération de texte. J'ai donc décidé d'utilisé un autre modèle (toujours basés sur l'architecture Transformers et présentant de nombreuses similitudes avec BERT) afin d'intégrer à mon pipeline une production de résumés et de traductions des articles de presse, le modèle BART.

4.4 La génération de résumés : Le modèle BART

En introduisant le modèle BERT, Jacob Devlin et son équipe ont décidé de ne pas prendre en compte les tâches de génération de texte et de laisser le décodeur de coté dans l'architecture des Transformers classiques. Le principe de bi-directionnalité de l'attention peut cependant s'avérer tout aussi pertinent pour une application à une tâche "Sequence-to-Sequence". C'est pour une telle application de ce nouveau mécanisme que l'équipe de recherche de Facebook AI, dirigée par Mark Lewis, publie le papier "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension" [5] en 2019.

4.4.1 Le modèle BART : Bidirectional Auto-Regressive Transformer

Le modèle BART est un modèle sequence-to-sequence basé sur une architecture Transformers classique 4. Cependant, Mark Lewis et son équipe ont décidé d'incorporer à cette architecture les principes introduits dans d'autres modèles. BART est donc constitué d'un encodeur bi-directionnel similaire au modèle BERT et d'un décodeur auto-régressif utilisé notamment par les modèles GPT. Pour le modèle BART-large, qui est celui que j'utilise dans mon étude, l'encodeur et le décodeur sont constitués de 12 couches similaires à celles

des Transformers.

Si l'encodeur présente de très fortes similitudes avec le modèle BERT, l'équipe de Facebook AI a décidé de modifier quelque peu l'entraînement. Quand les poids de BERT étaient pré-entrainés sur des tâches de Mask Language Model et de Next Sentence Prediction, le modèle BART est lui pré-entrainé sur des textes qui ont subis plus de modifications. L'équipe de Mark Lewis a effectivement décidé d'ajouter les mécanismes suivants :

- "Token Deletion" : Un mot aléatoire est supprimé et le modèle doit détecter la position manquante).
- "Text Infilling" : Plusieurs mots sont remplacés par des token [MASK].
- "Sentence Permutation" : L'ordre des phrases est mélangé dans le texte.
- "Document rotation" : Un mot est choisi au hasard et la phrase est tournée pour comencer par ce mot.

L'ensemble de ces modifications sont faites aléatoirement et peuvent être combinées lors de l'entrainement du modèle. Mark Lewis et son équipe estiment que ces tâches, plus compliquées que celles du modèle BERT classique, vont permettre d'optimiser un peu plus la capacité du modèle BART à capturer les informations contextuelles et sémantiques du texte.

Le décodeur est lui aussi inspiré du modèle Transformers mais les auteurs ont décidé de se référer au modèle génératif GPT-1 dans lequel la couche d'activation ReLu avait été remplacée par une couche GeLU, c'est l'unique différence d'architecture avec un décodeur classique. Ce décodeur génère du texte de façon auto-régressive, c'est à dire qu'il le génère mot par mot, de gauche à droite. Un mécanisme utilisé notamment dans le modèle Seq2Seq introduit en section 3.3.1.

Finalement, l'encodeur et le décodeur de BART sont entrainés simultanément sur une tâche de "Denoising". Le principe est de fournir une phrase tronquée en entrée de l'encodeur, ce dernier doit corriger la phrase afin de fournir une séquence cohérente au décodeur. Le dernier mot de cette séquence est supprimé et la séquence est fournie au décodeur qui s'entraîne à prédire le mot supprimé. L'objectif de l'entraînement est de minimiser la fonction de perte (ici une fonction d'entropie croisée) entre l'output du décodeur et l'input de l'encodeur.

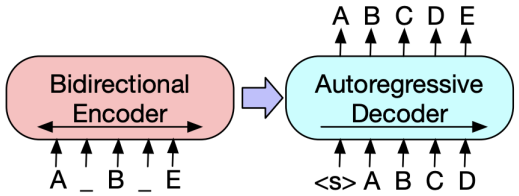


FIG. 7 – Entraiînement du modèle BART

Un tel entraînement permet donc d'obtenir un modèle Sequence-to-Sequence capable de capturer des relations contextuelles bidirectionnelles et de générer du texte de manière auto-régressive. Ce modèle pré-entrainé offre donc une possibilité de finetuning sur de nombreuses tâches de génération de texte telles que la production de résumés notamment.

4.4.2 La génération de résumés

Pour un outil d'analyse de news financières, la possibilité de générer des résumés d'article de presse est très importante. Cela permettra de fournir aux gérants de fonds les informations les plus pertinentes de chaque texte, pour chaque entreprise.

Cette tâche de summarization peut être traitée selon de nombreuses méthodes qui peuvent être regroupées en deux catégories. Les méthodes extractives et abstractives.
Les méthodes extractives sont les premières méthodes à avoir été mise en place pour une telle tâche. Elles consistent uniquement à extraire les informations les plus importantes du texte et ne nécessitent aucune génération de texte. L'objectif des modèles de summarization extractives est généralement d'extraire les phrases qui contiennent le plus d'informations sur le texte. Pour cela, de nombreuses méthodes ont été mises en place en se basant sur une hypothèse précise : les phrases qui présentent le plus de similitudes avec le reste du texte sont les phrases qui contiennent le plus d'information sur le texte à résumé.
Le premier modèle d'extraction de résumé est très basique et se base sur la fréquence d'occurences des mots dans un texte. C'est le modèe TF-IDF (Term Frequency-Inverse Document Frequency) introduit en 2009 [18]. Cette méthode consiste à calculer pour chaque mot W, le Term-Frequencey :

$$\mathrm{TF(W)} = \frac{\text{nombre d'occurence de W}}{\text{nombre de mots}}$$

et l'Inverse Document Frequency :

$$\mathrm{IDF(W)} = \log\left(\frac{\text{nombre de phrases}}{\text{nombre de phrases avec le mot W}}\right)$$

L'objectif est de créer deux matrices qui contiennent ces informations pour chaque mot. En les multipliant, on obtient donc un score d'importance pour chaque mot. Il est ensuite simple d'attribuer à chaque phrase un score d'importance en sommant les scores des mots qui constituent la phrase. En placant un seuil de scores, on peut générer un résumé en sélectionnant toutes les phrases au dessus de ce score.
Si cette méthode semble assez basique, l'introduction des modèles Transformers a permis d'apporter des améliorations importantes tout en gardant l'idée initiale. Les modèles Transformers peuvent effectivement être utilisés pour créer des embeddings de phrases. En conservant l'idée que les phrases les plus similaires au reste du texte sont les plus importantes, il suffit donc de créer une matrice de similarités entre les embeddings de chaque phrase. Cette matrice est généralement calculée selon la formule de similarité cosinus :

$$\cos(\theta) = \frac{Seq_1 . Seq_2}{\|Seq_1\| \, \|Seq_2\|} \tag{2}$$

A nouveau, un seuil de scores est appliqué et le résumé est généré en sélectionnant les phraes au dessus de ce seuil.

En permettant uniquement d'extraire des phrases d'un texte, les méthodes extractives paraissent limitées pour la génération de résumé. Lorsqu'un humain doit générer un résumé, il capture les informations importantes du texte et génère un texte qui condense ces informations. C'est sur un tel mécanisme que se basent les méthodes abstractives. Contrairement aux méthodes extractives, les modèles de summarization abstractive doivent avoir la capacité de générer du texte.
A nouveau, l'architecture Encoder-Decoder semble donc nécessaire pour une telle problèmatique et les modèles Transformers ont permis d'obtenir de très bonnes performances pour une telle tâche. Le mécanisme de génération abstractive de résumés est en effet très similaire à une tâche sequence-to-sequence classique.

L'encodeur génére une représentation vectorielle des informations principales contenues dans le texte et le décodeur prend cette représentation en entrée pour générer une phrase à partir de ces informations. Le modèle BART, composé d'un encodeur bidirectionnel et d'un décodeur auto-régressif représente notamment l'état de l'art dans ce domaine. En fine-tunant le modèle sur le dataset DailyMail/CNN, les auteurs du papier [5] ont en effet réussi à obtenir les meilleures performances sur la tâche de summarization.

4.5 La traduction de texte : Le modèle T5

Comme l'ensemble des Transformers, les modèles BERT et BART utilisent le principe de Transfer Learning. Les connaissances acquises pendant le pré-entrainement sont utilisées et adaptées à des tâches spécifiques durant le fine-tuning. Selon les tâches à traiter, le fine-tuning de ces modèles nécessite une modification des dernières couches du modèle et les paramètres d'entrainement peuvent différer.
En 2020, une équipe de recherche de Google AI s'est intéressée à un principe permettant de créer un modèle capable de traiter une grande partie des tâches de NLP sans avoir à modifié l'architecture ou l'entrainement du modèle. Ce principe est présenté par Colin Raffel et son équipe dans le papier "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" [12] qui introduit le modèle t5.

4.5.1 Le modèle T5 et le mécanisme de Transfer-Learning

L'intuition sur laquelle se basent les auteurs du papier [12] est que l'ensemble des tâches de NLP peuvent être considérées selon la même structure qui consiste à fournir une phrase d'entrée au modèle pour que ce dernier génère une phrase de sortie. Le modèle doit simplement pouvoir comprendre la tâche qui lui est demandée en capturant un contexte au sein de la phrase d'entrée.
En se basant sur cette idée, Colin Raffel et son équipe ont donc créé un modèle basé sur l'architecture Transformers et constitué de douze couches d'encodeurs et douze couches de décodeurs, le modèle T5. Ce modèle "Texte-to-Texte" permet de traiter toutes les tâches de NLP sans en modifier l'architecture, les hyper-paramètres et les paramètres d'entrainement. Le modèle nécessite simplement d'ajouter un préfixe à chaque phrase pour spécifier la tâche à effectuer. Un préfixe spécifique est attribué à chacune des tâches de NLP. Si l'on veut traduire la phrase "Apple beats estimates", il suffit de fournir au modèle la phrase "translate English to French: Apple beats estimates". Le papier [12] introduit également le dataset C4 (Colossal Clean Crawled Corpus), une base de données constituée de 750 GB de données textuelles nettoyées et qui représente une importante base d'entrainement non-supervisé.

La principale motivation de l'équipe de Google AI était de créer un modèle permettant une analyse comparative des méthodes de Transfer Learning. Ce modèle étant en capacité de traîter toutes les tâches de NLP avec la même architecture, il permet de comparer diverses stratégies d'entrainement sur un benchmark très important. Pour l'entrainement non-supervisé par exemple, les auteurs ont conclu qu'une approche efficace consistait à reprendre l'approche de Mask Modelling des modèles BERT en masquant cette fois plusieurs mots au sein de la même phrase. Un pré-entrainement plus long sur une base de données plus importante que les entraînements des Transformers classiques est également apparu comme une source d'amélioration pour le modèle.
Surtout, ce modèle permet d'introduire l'entraînement multi-task qui consiste à pré-entrainer le modèle sur l'ensemble des tâches à étudier. Le principe est de concaténer l'ensemble des datasets d'entraînement supervisé conçus pour chaque tâche, de modifier les phrases d'entrées pour que le modèle puisse comprendre la tâche attendue et d'entraîner le modèle sur cette nouvelle base de données. Ce mécanisme d'entraînement suppervisé ne suffit pas pour obtenir des résultats comparables à l'approche classique qui consiste à

effectuer un entraînement général non supervisé avant de fine-tuner le modèle sur des tâches spécifiques. Cependant, des performances vraiment satifsfaisantes sont obtenues en ajoutant une étape de fine-tuning à cet entraînement multi-task.

En entrainant le modèle T5 sur le corpus C4, les auteurs de ce papier ont donc introduit un modèle de langage efficace et fine-tunable sur tout type de tâches grâce à une architecture nouvelle. Ce modèle a permis d'établir une étude comparative des différentes approches d'entrainement des modèles de NLP mais également d'obtenir des performances proches de l'état de l'art sur un benchmark de tâches très diversifiées. En chargeant depuis la librairie HuggingFace les poids du modèle T5 finetunés sur une tâche de traduction et en modifiant les phrases d'entrée pour introduire la tâche à effectuer, je suis donc en mesure d'effectuer une traduction efficace des résumés générés par le modèle BART.

4.5.2 La génération de traductions

Comme énnoncé en partie 3.1.1, la tâche de traduction représente le premier défi que s'étaient lancés les chercheurs dans le domaine du NLP en 1954, avec l'algorithme IBM pour la traduction de 60 phrases entre l'anglais et le russe.
Les vingt premières années de recherche sur le sujet ont permis d'introduire de nombreux modèles basés sur des dictionnaires bilingues et de règles lingustiques fournies préalablement à la machine. Si ces modèles ont permis de générer des premières traductions automatiques, leur utilisation était limitée par le fait qu'ils nécessitaient des bases de données importantes et mises à jour continuellement, ces modèles n'étaient pas capables de s'adapter en cas de mots ou de règles non présents dans les dictionnaires.

Les modèles statistiques introduits en NLP durant les années 1990 ont permis d'envisager de nouvelles méthodes de traduction automatique, c'est l'introduction des SMT (Statistical Machine Translation). Le principe de ces modèles est d'utiliser des méthodes de Machine learning sur des "corpus parrallèles" afin de capturer des relations statistiques entre deux langues données. Un corpus parrallèle est un corpus de texte contenant les mêmes phrases dans deux langues différentes.
Les premiers modèles statistiques se concentraient uniquement sur les mots qui constituent la phrase ("Word-based SMT"). Cette approche repose sur deux composantes importantes. Tout d'abord, la composante lexicale. Le modèle capture les co-occurences entre les mots dans chaque langue en analysant le nombre de fois où un mot de la phrase source se retrouve dans les phrases de la langue cible. Cela peut permettre au modèle de prédire la traduction d'un mot avec une formule de maximum de vraisembance par exemple. La deuxième composante est l'alignement des mots, ce mécanisme consiste à effectuer un mapping entre les mots d'une séquence source et d'une séquence cible. Ce mapping permet au modèle de capturer les correspondances entre les mots. La fonction d'alignement n'est pas forcèment bijective, le mot "implemented" peut être mappé avec les mots "mis en application", ce qui permet au modèle de comprendre que la traduction n'est pas forcèment word-to-word.
Mais les modèles basés sur des phrases (ou des séquences de mots) semblent plus pertinents pour la traduction. On ne peut traduire correctement une langue en traduisant simplement chaque mot, il faut capturer les relations syntaxiques entre les deux langues. L'approche basée sur des séquences ("Phrase-Based SMT") a donc été introduite pour améliorer ces traductions automatiques. Pour cela, les Phrase-Based models prennent des paires de phrases et les segmentent en plusieurs séquences. Le but est ensuite d'aligner ces séquences entre elles. Cela permet de capturer un sens plus global au sein de la phrase ainsi que des relations syntaxiques. C'est une telle approche qui fût utilisée lors de la création du premier modèle Google Translate en 2006.

Si ces approches statistiques ont permis d'améliorer grandement les performances des modèles de traduction, les modèles de deep learning basés sur les architectures Transformers ont encore une fois permis d'obtenir des résultats nettement supérieurs. Comme pour la génération de résumé, la traduction est une tâche Sequence-to-Sequence classique et les modèles dotés d'encodeurs et décodeurs semblent les plus pertinents pour une telle tâche. En finetunant T5 sur un dataset de traduction, les auteurs du papier [12] ont encore une fois obtenu une performance vraiment satisfaisante.

En publiant le papier "Attention is all you need" [1] et en introduisant les modèles Transformers, les chercheurs de Google AI ont ouvert la porte à des progrès exceptionnels dans le domaine du NLP. De nombreux modèles tels que BERT pour l'analyse de langage, BART pour la génération de textes et T5 dans l'ensemble des tâches, se sont inspirés de cette architecture Transformers afin de l'améliorer et de l'adapter. Etant pré-entrainés sur des corpus de textes presque illimités, ces modèles offrent des possiblilités de fine-tuning sur une grande diversité de tâches. Dans le cadre d'une analyse de news financières et la création d'un outil d'aide à la décision, ces modèles représentent donc une solution vraiment pertinente.

5 FinBERT-ABSA : L'analyse de news financières

5.1 Introduction

Je vais désormais introduire mon travail de recherche : Le modèle FinBERT-ABSA. En me basant sur les modèles et les mécanismes introduits précédemment, j'ai décidé de créer un modèle d'analyse de langage qui a pour objectif d'extraire les entreprises mentionnées dans les titres de news financières et de leur attribuer un score ESG calculé grâce au contexte que fournit la phrase à son sujet.

5.1.1 Le modèle FinBERT

Pour cela, le modèle BERT représente une base de travail logique à la vue des caractéristiques introduites en section 4. Plus spécifiquement, j'ai décidé d'étudier un modèle BERT déjà fine-tuné sur un dataset financier : le modèle FinBERT.
FinBERT est un modèle de classification de sentiment introduit par Dogu Tan Araci qui publie le papier "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models" [14] en 2019.
Pour adapter ce modèle, l'auteur a décidé d'utiliser deux stratégies et d'en comparer les résultats. La première méthode consiste à pré-entrainer à nouveau le modèle BERT sur un dataset financier, le modèle est donc entraîné à prédire les mots masqués dans une phrase ou les phrases suivantes comme lors de l'entraînement initial de BERT. La deuxième méthode consisite simplement à fine-tuner les poids sur une tâche de classification de sentiment appliquée à un dataset financier. Ces deux méthodes ont finalement permis d'obtenir des résultats presque similaires et le modèle FinBERT a obtenu des meilleurs performances que l'ensemble des modèles existants pour la tâche de classification de sentiments sur le dataset financier "Financial PhraseBank". Ce modèle constitue l'état de l'art dans le domaine de la compréhension de langage financier et représente donc un bon point de départ pour la construction de mon modèle.

En disposant d'un modèle de compréhension de langage entrainé sur des données financières, il me suffit donc de l'adapter afin de prendre en compte le caractère ESG dans la classification de sentiment et d'en modifier l'architecture pour répondre à une tâche d'Aspect-Based-Sentiment Analysis (ABSA).

5.1.2 Idée d'adaptation du modèle

L'enjeu de mon travail de recherche est donc de calculer un score ESG pour une entreprise en capturant les informations d'un titre de news financières.
Pour la résolution d'une telle tâche, de nombreuses caractéristiques du modèle BERT semblent pertinentes et ont motivé mon choix de ce modèle FinBERT.

Les modèles basés sur l'achitecture BERT sont en effet entraînés pour fournir des embeddings de 768 valeurs pour chaque tokens de la séquence prise en entrée.
Le mécanisme présenté en section 4 pour fournir ces embeddings consiste d'abord à générer les embeddings de chaque tokens qui ont été crées pendant le pré-entrainement du modèle. Avant d'alimenter la première couche du modèle, ces embeddings ne contiennent que les informations sémantique sur le mot, sa définition. Ces embeddings sont ensuite fournis au modèles qui effectue les opérations présentées en section 4 afin de les mettre à jour et de fournir des représentations vectorielles contenant les informations pertinentes pour la réolution de la tâche sur lequel le modèle à été pré-entrainé.

C'est cette particularité qui motive mon choix du modèle FinBERT et mon idée d'adaptation pour la tâche ABSA. Le modèle FinBERT a en effet été entrainé pour la classification de sentiments sur des données financières. Les poids du réseau de neurones sont donc optimisés pour capturer les informations sur le sentiment de la phrase et fournir des embeddings qui prennent en compte cette information. Les vecteurs générés par le modèle pour chaque token contiennent les informations contextuelles (et donc le sentiment) que fournit la phrase par rapport à ces tokens.
En extrayant les embeddings correspondant aux tokens de l'entreprise mentionnée dans la phrase, il est donc possible que ces vecteurs contiennent les informations nécessaires pour la classification de sentiment par rapport à cette entreprise.

Pour appuyer cette hypothèse et confirmer le fait que les embeddings générés par le modèle FinBERT pour chaque mot contiennent bien les informations contextuelles de la phrase, j'ai décidé d'étudier deux phrases présentes dans mon dataset : "Italy competition body starts probe against apple" et "Microsoft Sees Biggest Rally of the Year After Sales Top Estimates".
Ces deux phrases expriment des sentiment différents par rapport aux entreprises qui y sont mentionnées et les embeddings générés par la dernière couche de FinBert pour ces deux entreprises ne devraient donc pas présenter trop de similarités. De plus, les embeddings correspondant aux mots "Apple" et "Microsoft" que le modèle va fournir à la première couche du réseau de neurones devraient également être différents même si il s'agit de deux entreprises plutôt similaires.
J'ai donc décidé de copier chacune de ces phrases en échangeant les entreprises. Je fournis donc au modèle 4 phrases : "Italy competition body starts probe against apple", "Italy competition body starts probe against Microsoft", "Microsoft Sees Biggest Rally of the Year After Sales Top Estimates", "Apple Sees Biggest Rally of the Year After Sales Top Estimates".
Le but est d'isoler les premiers embeddings générés avant de rentrer dans le réseau de neurones ainsi que les embeddings générés par la dernière couche du modèle afin de les comparer. J'ai isolé ces embeddings en récupérant l'index des tokens correspondants à ces deux entreprises et en extrayant les vecteurs situés à cet index dans la première et la dernière couche du modèle. Pour l'entreprise Microsoft qui est séparée en deux tokens, il a fallu effectuer un pooling entre les deux vecteurs, j'ai décidé de prendre la moyenne (une pratique courante pour le pooling).

L'algorithme t-SNE

La réprésentation graphique de vecteurs de dimension 768 étant impossible, j'ai décidé de leur appliquer l'algorithme t-SNE (t-distributed Stochastic Neighbor Embedding) afin de réduire l'analyse à des vecteurs de deux dimensions. L'algorithme t-SNE est une technique de réduction de dimension introduite par Laurens van der Maaten et Geoffrey Hinton en 2008 [17]. Le principe est de modéliser les similarités entre les points d'un espace de grande dimension pour les représenter dans un espace en deux ou trois dimensions. L'algorithme calcule d'abord la probabilité entre chaque points $x_i et x_j (i \neq j)$ d'être "voisins" selon une distribution gaussienne centrée en x_i :

$$p_{i|j} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

avec σ_i la variance de la distribution gaussienne définie au prélable.
Une fois ces probabilitées calculées, l'algorithme t-SNE représente ces similarités entre chaque point dans un espace de deux dimensions selon la formule :

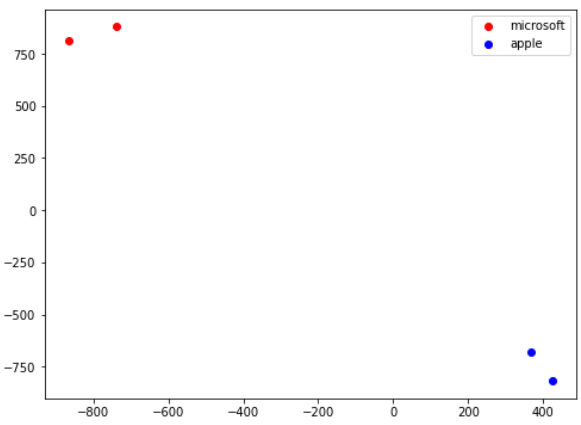
$$q_{ij} = \frac{(1 + |y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + |y_k - y_l\|^2)^{-1}}$$

Cette formule permet de modéliser la distance entre les points, l'emplacement des points sur le graphique se calcule alors en minimisant la distance de Kullback-Leibler entre les distribution p et q définies par :

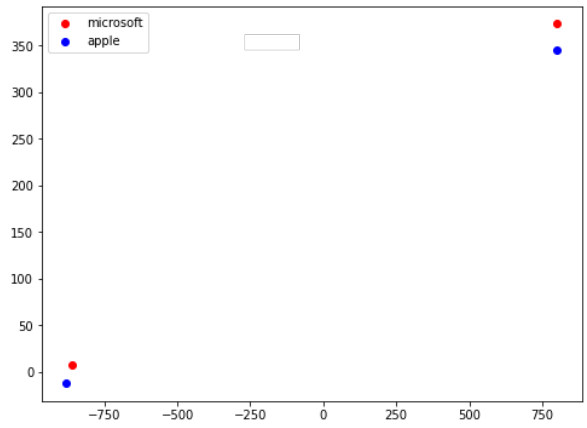
$$\mathrm{KL}\,(\mathrm{P} \,\|\, \mathrm{Q}) = \sum_{i \neq j} p_{ij} \log(\frac{p_{ij}}{q_{ij)}}$$

Représentation graphique des embeddings

En appliquant cet algorithme t-SNE aux 4 embeddings générés initialement par FinBERT puis aux 4 embeddings générés par la couche finale du modèle, on obtient finalement les représentations graphiques suivantes des entreprises dans chaque phrase.



(a) Embeddings générés initialement par FinBERT



(b) Embeddings générés par la dernière couche de FinBERT

FIG. 8 – Représentation t-SNE des embeddings correspondant aux entreprises Apple et Microsoft

On s'apercoit que les embeddings pré-entrainés par le modèle FinBERT et générés initialement pour chaque entreprises sont complétement différents de ceux issus de la dernière couche.
Logiquement, les représentation vectorielles initiales de l'entreprise Apple sont très similaires puisque le modèle associe simplement les tokens "Apple" au même vecteur. Le léger écart se justifie par le fait que le

modèle ajoute un encodage de position au sein de la phrase, le mot “Apple” est en première position dans l’une des phrases et en dernière position dans l’autre. Le même raisonnement est valable pour Microsoft.
La représentation t-SNE des embeddings issus de la dernière couche est totalement différente, on s’aperçoit que ce sont les entreprises issues phrases similaires qui ont été sélectionnées comme voisines. L’embedding du mot “Apple” issu de la phrase “Italy competition body starts probe against apple” semble très similaire à celui du mot “Microsoft” issu de la phrase “Italy competition body starts probe against Microsoft”.

Ces représentations graphiques semblent valider l’hypothèse faite sur la génération des embeddings par le modèle FinBERT. Les embeddings fournis par la dernière couche du modèle contiennent certes des informations sémantiques sur les mots étudiés (ce qui explique les légers écarts sur la représentation t-SNE), mais ces vecteurs semblent surtout capturer les informations contextuelles autour de ce mot. Deux mots différents placés au même endroit d’une même phrase possèdent des vecteurs vraiment similaires. Le modèle FinBERT étant entrainé à predire un score de sentiment, ces informations contextuelles contiennent très probablement des informations sur le sentiment de la phrase par rapport aux entreprises.
En entrainant mon modèle à capturer ces informations, je devrais donc être en mesure d’effectuer une analyse de sentiments par rapport aux entreprises mentionnées dans les titres de news financières.

5.2 Le Dataset

Pour adapter mon modèle et l’entrainer sur des données financières, nous avons décidé de faire appel au groupe américain Bloomberg. Bloomberg publie chaque jour des dizaines de milliers d’articles de presse dans le monde et propose une solution aux entreprises pour collecter toutes ces données à l’aide d’une API. Les bases de données récupérées contiennent les titres et le contenu des news mais également les dates de publications et de nombreuses annotations, notamment sur les thèmes abordés dans chaque article. Pour répondre à ma problématique, j’ai donc sélectionné les articles traitant de sujets ESG. En sélectionnant uniquement les articles en anglais, cela représentait une cinquantaine de news par jour.

Nous sommes partis du principe que les titres de news condensaient plutôt bien les informations contenues dans les articles (surtout pour une analyse de sentiments) et avons décidé d’étudier uniquement ces titres pour notre problématique. Ces données étaient en effet nettement plus propres que le corps entier de chaque article et facilitaient réellement l’analyse de sentiment.
Cependant, l’entrainement de mon modèle nécessitait des données labellisées et ne pouvait prendre en compte que les titres de news mentionnant des entreprises. J’ai donc constitué une base de données comportant les titres de news traitant du sujet ESG entre les mois de décembre et avril et labellisé à la main l’ensemble de ces titres. La labbelisation de ces données consistait à extraire de chaque titre une entreprise et un score de sentiment (Négatif = -1, Neutre = 0, Positif = 1). Pour les titres mentionnant deux entreprises, j’ai décidé de dupliquer la phrase et d’effectuer cete labellisation pour chaque entreprise. J’ai finalement obtenu un dataset constitué de 2326 titres de news et comportant trois colonnes : Titre, Entreprise et Sentiment. C’est ce dataset qui sera utilisé pour l’entrainement et l’évaluation de mon modèle FinBERT-ABSA.

5.3 Le modèle

Le modèle FinBERT étant déjà pré-entrainé à comprendre des textes financiers, le dataset constitué de 2326 titres est suffisant pour le finetuning de ce modèle si l’on conserve les poids du modèle pré-entrainé. Nous disposons donc des ressources nécessaires pour mettre en place un modèle capable de répondre à notre problématique. Comme énnoncé en section 4.2.3, je vais traîter cette tâche “End-to-End ABSA” en

divisant le problème en deux parties. Je vais tout d'abord mettre en place une fonction de recconaissance d'entreprise avant de modifier l'architecture du modèle FinBERT pour attribuer un sentiment à l'aspect de la phrase que constitue cette entreprise.
Les modèles que je vais utiliser sont mis à disposition par HuggingFace qui donne la possibilité de télécharger des milliers de modèles de NLP à l'aide de la librairie Transformers. Cette librairie permet notamment d'utiliser les poids pré-entrainés ainsi que les fonctions de tokenization de ces modèles.

5.3.1 NER pour la détection d'entreprise

La première partie de mon travail consiste donc à extraire les entreprises mentionnées dans les phrases étudiées. C'est une tâche de Name-Entity Recognition que l'on va adapter pour extraire uniquement les entités correspondant à des entreprises.

Pour se faire, nous allons utiliser l'un des modèles qui représente l'état de l'art dans cette tâche de NER, le modèle RoBERTa. Ce modèle a été introduit par l'équipe de recherche de Facebook AI dans le papier "RoBERTa: A Robustly Optimized BERT Pretraining Approach" [13] avec pour objectif d'optimiser le modèle BERT pour certaines tâches d'analyse de langage (notamment le NER).
Pour cela, les auteurs de ce papier ont décidé de conserver l'architecture BERT mais de modifier le pré-entrainement en supprimant la tâche de Next Sentence Prediction et en entrainant le modèle sur une base de données dix fois plus importante que la base d'entrainement de BERT. Certains hyperparamètres comme les tailles de batchs et de séquence ont également été modifiés et le modèle obtient finalement des meilleures performances que BERT sur de nombreux benchmarks.

A l'aide de la librairie HuggingFace, j'ai donc importé le modèle RoBERTa finetuné sur la tâche NER, nommé "roberta-large-ner-english". Les performances de ce modèle sont très satisfaisantes pour la recconnaissance d'organisation (96,27% d'accuracy sur le dataset conl2003) et ne nécessite donc aucun finetuning particulier pour extraire les entreprises mentionnées dans les phrases. Par soucis de simplicité, l'algorithme d'extraction d'entreprises que je met en place ne prend en compte que les deux premières entreprises détectées par le modèle dans la phrase. Lorsque l'algorithme détecte deux entreprises, la phrase est dupliquée et annotée pour chacune des entreprises.
Une fois que les entités de chaque phrase ont été extraites par le modèle, un léger post-processing est nécessaire pour s'assurer que le modèle d'analyse de sentiment puisse matcher correctement l'entité extraite avec son emplacement dans la phrase initiale. Le modèle FinBERT utilise effectivement un tokenizer différent de celui de RoBERTa, ce qui implique que nous devons lui fournir des entités sous forme de mots et non de tokens générés par le NER. J'ai donc appliqué une fonction à l'ensemble des entités extraites pour supprimer les espaces et les majuscules qui peuvent apparaître lors de la détokenisation des entreprises détectées par le NER.

5.3.2 Pré-processing des données

Cet algorithme de recconaissance d'entreprise fournit finalement un dataset contenant trois colonnes : La date, le titre de l'article et l'entreprise détectée. Mais ce dataset doit être adapté à un format compatible avec le modèle que je vais créer et doit surtout comporter plus d'informations. Un pré-processing est donc nécessaire avant l'utilisation de mon modèle.

Tout d'abord, le dataset obtenu est sous forme de dataframe pandas. Cette architecture n'est pas optimale

pour l'usage de la librairie PyTorch et ne peut être transférée sur un GPU, ce qui rend impossible l'entrainement de mon modèle. Pour résoudre cela, j'ai donc converti mon dataframe en un Dataset. Les Datasets sont des architectures introduites par TensorFlow et optimisées pour travailler avec les librairies TensorFlow et PyTorch. Ce Dataset contient les mêmes données que le dataframe initial mais les données textuelles sont transformées en tenseurs. J'ai également ajouté un padding sur les phrases d'entrée pour faciliter les calculs. Le mécanisme de padding consiste à générer des tenseurs nuls à la fin des phrases pour que celles-ci soient toutes de même taille. Cette opération est nécessaire pour l'usage des modèles Transformers.

L'usage de mon modèle nécessite que ce Dataset contienne de nouvelles informations sur les phrases et les entités qui en sont extraites. Afin de rendre possible la sélection des embeddings correspondants aux entreprises, il a fallu ajouter des colonnes contenant les informations sur la longueur de la phrase et des entités. La longueur d'un mot correspond au nombre de tokens (et donc de tenseurs) qui composent ce mot. Lors de la tokenization effectuée par les modèles BERT, un token de début '[CLS]' et de fin '[SEP]' de phrase est généré. Il a donc suffi de compter le nombre de tokens entre ces tokens de délimitation pour générer les longueurs de chaque phrase. Toutes ces informations ont logiquement été générées sous forme de tenseurs et inclues dans mon Dataset.

Mais le modèle créé doit surtout connaître l'emplacement des tokens correspondants à l'entreprise au sein de la phrase. J'ai pour cela utilisé la fonction *torch.eq* de la librairie Pytorch. Cette fonction a été implémentée pour effectuer une comparaison élément par élément entre deux tenseurs et renvoie un tenseur indiquant les emplacements de la phrase qui contiennent les tenseurs correspondants à l'entreprise. Cette fonction renvoie donc un tenseur contenant ces informations pour toutes les phrases qui constituent mon Dataset, une phrase correspondant à une 'ligne' de ce tenseur.
Malgré le post-processing effectué sur les données générées par le modèle NER, certaines entreprises ne sont pas retrouvées dans la phrase d'origine, ce qui est dû à une différence de tokenization par le tokenizer FinBERT. Cependant, le post-processing m'a permis de limiter cela au maximum et un peu moins d'1% des phrases présentent ce problème. J'ai donc décidé de supprimer les lignes correspondant à ces phrases dans l'ensemble des tenseurs de mon Dataset.
Enfin, cette fonction *torch.eq* renvoie parfois plusieurs emplacements au sein de la même phrase (quand l'entreprise est mentionnée plusieurs fois notamment). J'ai décidé de ne conserver que le premier emplacement indiqué pour chaque phrase.

Cet algorithme de pré-processing appliqué sur le dataframe fourni par le modèle RoBERTa permet donc d'obtenir un Dataset contenant l'ensemble des informations pertinentes sur les phrases à étudier : Le texte, l'entreprise, la taille du texte et de l'entreprise et l'emplacement de l'entreprise au sein du texte. L'ensemble de ces informations sont contenues sous forme de tenseurs dans un Dataset afin de rendre possible l'entrainement du modèle sur un GPU. Les GPU (Graphics Processing Unit) sont des processeurs conçus pour effectuer rapidement des calculs massif en parrallélisant les opérations. L'usage d'un tel processeur s'est avéré nécessaire pour l'entraînement de mon modèle.

5.3.3 L'architecture du modèle

A partir des informations fournies par le pré-processing de mes données, j'ai finalement été en mesure de créer le modèle "FinBERT-ABSA". J'ai pour cela créé une classe composée d'une fonction *forward*. Cette fonction prend en entrée les données du Dataset issu du pré-processing et génère un score de sentiment pour chaque phrase par rapport aux entreprises qui y sont mentionnées. Pour permettre le calcul sur GPU et

optimiser l'entraînement et l'éxécution de mon modèle, l'ensemble des opérations sont effectuées sur des batchs de tenseurs. Les batchs sont des groupes de données qui seront traitées simultanément par le modèle. Dans mon étude, je considère des batchs de 128 observations.
La fonction *forward* peut être décomposée en trois étapes qui permettent la génération de ce score de sentiment : La génération d'un masque d'attention correspondant aux emplacements des entreprises dans les phrases, l'utilisation de ce masque sur les embeddings fournis par le modèle FinBERT initial et la classification des embeddings obtenus.

Génération d'un masque d'attention

Le modèle FinBERT est entrainé pour générer un score de sentiment sur des phrases financières. Il est construit selon l'architecture BERT classique à laquelle on ajoute une couche de classification appliquée aux embeddings générés par la dernière couche de l'encodeur. Ce sont ces embeddings qui sont pertinents dans le cadre de mon étude.
En appliquant le modèle FinBERT aux batchs de mon Dataset, la dernière couche d'embeddings est de la forme (batch size, sequence size, embeddings size). Ayant effectué un padding lors du pré-processing de mes données, toutes les phrases sont constitués de 64 tokens, les embeddings sont des vecteurs de dimension 768 selon l'architecture BERT. La couche d'embeddings étudiée est donc de la forme (128, 64, 768).
Afin d'appliquer un masque d'attention à cette couche, je dois donc créer une matrice en trois dimensions, de forme (128, 64, 1).
La première étape consiste à créer un tenseur de dimension (128,64) et des tenseurs pour stocker les indices de lignes et de colonnes.

```
matrix = torch.zeros(num_elements, max_index)
row_indices = torch.arange(num_elements).unsqueeze(1)
column_indices = torch.arange(max_index).unsqueeze(0)
```

Ces tenseurs ne contiennent alors que des 0 mais le code suivant est utilisé pour stocker les indices de colonnes qui doivent être remplacés par un 1.

```
mask = (column_indices >= indexes_entities_in_phrases.unsqueeze(1))
& (column_indices < indexes_entities_in_phrases.unsqueeze(1)+input_length_new2.unsqueeze(1))
```

Les variables "Indexes entities in phrases" et "input length new2" représentent respectivement l'emplacement et la taille des entreprises au sein des phrases. Le code fourni remplace donc logiquement les indices correspondant à l'emplacement de tout les tokens représentant l'entreprise au sein de la phrase par un 1.
L'une des particularités de la programmation en tenseurs est que cette fonction s'applique à chaque observation du batch sans avoir à le préciser, ces indices de colonne sont donc stockés pour chaque ligne du tenseur. Ils sont finalement appliqués au tenseur *matrix* afin d'obtenir un masque d'attention pour les emplacements des tokens des entreprises. L'objectif étant d'obtenir un tenseur de trois dimensions pour l'appliquer à notre couche d'embeddings, on utilise la fonction *unsqueeze* pour ajouter la toisième dimension.

```
matrix[row_indices, column_indices] = mask.float()
mask_matrix = matrix.unsqueeze(-1)
```

On obtient donc un tenseur de même forme que la dernière couche d'embeddings du modèle FinBERT. Ce tenseur va nous permettre d'extraire uniquement les embeddings correspondant aux entreprises que l'on veut étudier.

Extraction des embeddings correspondant aux entreprises

La seconde étape de cette fonction *forward* consiste donc à appliquer ce masque d'attention à la dernière couche d'embeddings. Pour cela, mon modèle doit d'abord récupérer cette couche d'embeddings en appelant le modèle FinBERT sur les phrases constituant le batch étudié et en utilisant l'attribut *last-hidden-state* des classes des modèles BERT. Cet attribut permet de récupérer la dernière couche d'embeddings :

```
outputs = self.bert(input_ids=input_phrases_new, output_hidden_states=True)
last_hidden_state = outputs.last_hidden_state
```

Une fois cette couche récupérée, le modèle doit y appliquer le masque d'attention. On utilise pour cela le produit matriciel de Hadamard qui consiste en une multiplication terme à terme de deux matrices de même dimensions.
Formellement, si l'on a deux matrices $A,B \in \mathbb{C}^{m\times n}$, le produit de Hadamard est une matrice $A.B \in \mathbb{C}^{m\times n}$ dont les coefficents sont $(A.B)_{i,j} = (A)_{i,j} \times (B)_{i,j}$.
On dispose en l'ocurrence de tenseurs de trois dimensions, les deux premières dimensions sont identiques et la multiplication terme à terme se fera donc entre celles-ci. La dernière dimension ajoutée au tenseur de masque implique simplement que chaque embeddings de la dernière couche gardera sa dimension de 768. Le shéma ci-dessous représente ce mécanisme de multiplication terme à terme de manière simlpifiée en supposant batch size = 2 et sequence size = 7. Chaque token correspond à l'embedding de 768 valeurs généré par FinBERT pour ce token et le masque d'attention est celui généré par la première étape du modèle pour ce batch :

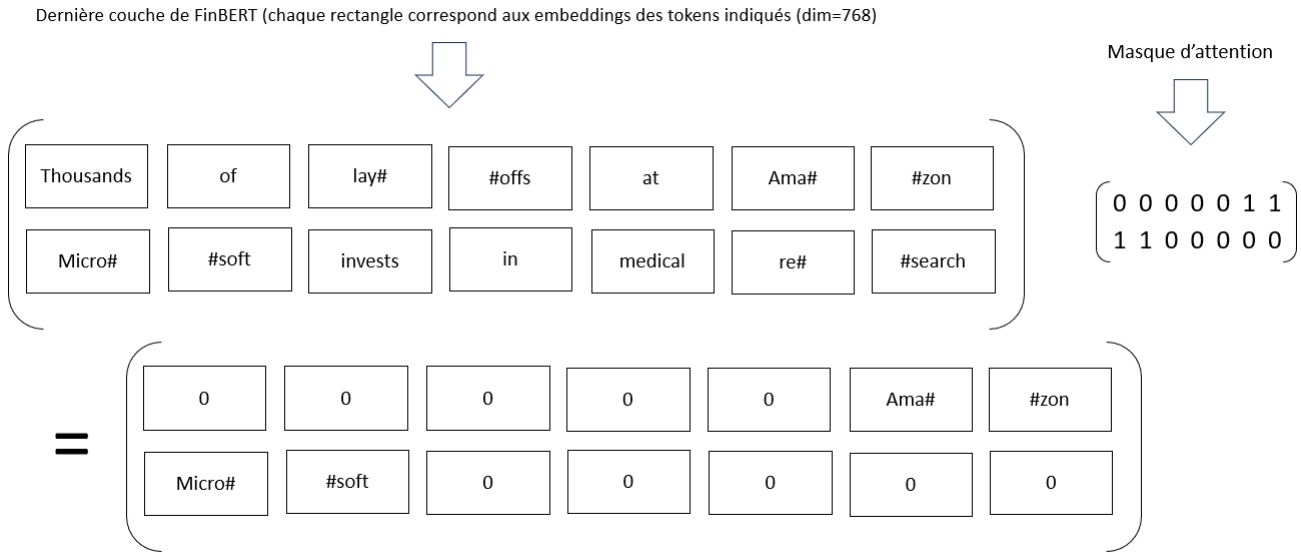


FIG. 9 – Multiplication terme à terme pour l'application du du masque d'attention

On effectue donc cette multiplication de Hadamard entre la dernière couche et le masque d'attention à l'aide de l'opérateur (*) :

```
entity_hidden_state = last_hidden_state*mask_matrix
```

Cela génère un tenseur de dimensions (128, 64, 768) dans lequel, pour chacune des 128 phrases, tout les embeddings sont constitués de 0 sauf ceux correspondants aux tokens des entreprise. Pour chaque phrase, on effectue un pooling entre les embeddings afin d'obtenir une unique représentation de l'entreprise qui y est mentionnée. C'est généralement la moyenne qui est utilisée dans de tels cas. On normalise ensuite l'embedding obtenu pour simplifier la classification que nous allons effectuer.

```
entity_embedding = entity_hidden_state.mean(dim=1)
entity_embedding = torch.nn.functional.normalize(entity_embedding)
```

Cette étape terminée, on obtient finalement un tenseur de dimensions (128, 768) que le modèle va utiliser pour la classification de sentiments. Ce tenseur est en effet composé d'un embedding par phrase du batch et cet embedding contient les informations contextuelles de la phrase par rapport à l'entreprise qui y est mentionnée.

Génération d'un score de sentiment

La dernière étape de mon modèle consiste à effectuer une classification sur chacun de ces embeddings afin de générer un score de sentiment. Après avoir testé de nombreuses architectures différentes, il s'est avéré que la plus efficace pour mon problème de classification était simplement constituée de deux couches linéaires que je crée au préalable dans l'initialisation de mon modèle:

```
self.linear_layer = nn.Linear(768, 512)
self.linear_layer_bis = nn.Linear(512, 3)
```

Afin d'éviter l'overfiiting et de permettre au modèle de capturer des relations non linéaires entre les composantes de l'embedding, des couchse de dropout et d'activations sont appliqués sur les sorties de la première couche linéaire:

```
output = self.linear_layer(entity_embedding)
output = torch.dropout(output, p=0.4, train=True)
output = F.relu(output)
```

Enfin, une fonction softmax est appliquée aux scores de sentiments attribués par la dernière couche linéaire afin d'obtenir des probabilités d'appartenir à chaque classe de sentiment (Négatif, Neutre ou Positif).

```
output = self.linear_layer_bis(output)
output = torch.softmax(output, dim = 1)
return output
```

Le modèle FinBERT-ABSA est donc constitué des douzes couches qui constituent un modèle BERT classique et de deux couches linéaires permettant la classification des sentiments. Ces deux dernières couches permettent au modèle de capturer les informations contextuelles contenues dans l'embedding représentant l'entreprise au sein de la phrase afin de générer un score de sentiment pour cette entreprise.

5.3.4 Entraînement et performance du modèle

Pour l'entrainement de ce modèle FinBERT-ABSA, nous disposons du dataset introduit en section 5.2 et constitué de 2326 titres de news labellisés avec une entité et un sentiment pour cette entité.
Le modèle de Name-Entity Recognition ne nécessite pas de ré-entrainement mais pour m'assurer que le modèle que j'ai créé pour l'analyse de sentiment détecte bien l'entreprise au sein de la phrase, je dois tout de même utiliser le modèle RoBERTa sur cette base d'entraînement afin de vérifier que les entités extraites coïncident avec les entreprises que j'ai labellisé au préalable.
Pour cela, j'ai appliqué une fonction aux entreprises que j'avais annotées et à celles extraites par le modèle afin de supprimer les espaces et les majuscules qui pourraient fausser l'analyse. J'ai ensuite décidé de ne conserver que les phrases pour lesquelles la chaîne de caractères extraite par le modèle était contenue dans celle que j'avais labellisée. Si le modèle extrait deux entités, je conserve celle qui coïncide avec ma labellisation. Sur les 2326 titres dont je disposais, le modèle de NER a extrait la bonne entité de 1900 phrases. J'ai décidé de conserver ces 1900 phrases pour l'entrainement de mon modèle et de supprimer les autres.
Je dispose donc d'un dataset de 1900 observations contenant la phrase, l'entreprise extraite et le sentiment par rapport à cette entreprise que j'ai labellisé au préalable. J'applique à ce dataset les fonctions de pré-processing introduites en section 5.3.2 et j'obtiens finalement un Dataset compatible avec le modèle FinBERT-ABSA que je souhaite entrainer.
Pour l'entrainement, je sépare ce Dataset en trois parties : Train, Validation et Test. Pour cela, je le sépare d'abord en un ensemble d'entrainement (80% du dataset) et un ensemble de test (20%), puis je crée l'ensemble de validation (20% du train). Je dispose donc d'un Dataset labellisé et pré-processé pour l'entrainement de mon modèle.

Avant de lancer cet entrainement, il faut cependant déterminer quelques paramètres et hyperparamètres pour en optimiser la performance.
Tout d'abord, l'ensemble d'entraînement dont je dispose est constitué d'environ 1400 observations. Cette quantité de données est négligeable par rapport aux corpus sur lesquels ont été pré-entrainé les modèles BERT et FinBERT et le fait de mettre à jour tout les poids du modèle FinBERT serait donc totalement contre productif. Surtout, ces poids sont pré-entrainés pour capturer un sentiment sur des données financières et sont par conséquent parfaitement adapatés à mon problème. J'ai donc décidé de geler ces poids lors de l'entraînement de mon modèle, seuls les poids des deux couches linéaires que j'ai ajoutées seront mis à jour. Ainsi, l'entrainement de mon modèle se concentre sur la classification des embeddings fournis par le modèle FinBERT initial. Ce mécanisme me permet d'entrainer mon modèle efficacement malgrè le peu de données à ma disposition, c'est tout l'intérêt des modèles pré-entrainés basés sur cette architecture Transformers.

Après un grid-search effectué manuellement pour détecter les hyperparamètres optimaux, j'ai décidé de sélectionner un taux d'apprentissage lr = 0.001 et un batch-size de 128. La fonction de perte utilisée est la fonction de Cross-Entropy categorielle. Cette fonction calcule les différences entre les labels et les probabilités prédites par le modèle et est considérée comme la fonction de perte à utiliser pour des réseaux de neurones constitués d'une couche d'output softmax pour la tâche de classification multi classes. La fonction de perte est alors calculée selon la formule :

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

avec C le nombre de classes et N le nombre d'obesrvations du Batch. Les labels sont hot-encodés dans ma fonction d'entrainement pour utiliser cette fonction de perte. J'ai également décidé d'utiliser l'optimiseur

Adam introduit par une équipe de recherche d'Open AI en 2014 [16]. Cet optimiseur se distingue notamment par ses capacités d'adaptation du taux d'apprentissage en fonction de l'historique des descentes de gradient et constitue aujourd'hui l'un des outils d'optimisation les plus utilisés pour l'entraînement des réseaux de neurones. Enfin, j'ai décidé d'entraîner mon algorithme sur 30 epochs. Je suis conscient que ce nombre peut paraître trop important mais le GPU me permet d'effectuer cet entraînement rapidement et j'ai ajouté un "early-stopping" permettant de stopper l'entrainement si la fonction de perte ne diminue plus sur l'ensemble de validation. Cela me permet notamment d'empêcher un possible overfitting de mon modèle sur les données d'entraînement.

Une fois ces hyperparamètres définis, le dataset est fourni à un DataLoader, un outil nécessaire pour l'entrainement de réseaux de neurones. Le Dataloader optimise en effet le Dataset à disposition pour l'entrainement, notamment en mélangeant les batchs à chaque epoch pour éviter le suraprentissage et en permettant au GPU d'effectuer les calclus en parrallèle. On entraine finalement notre modèle sur ce Dataset en utilisant un GPU Nvidia RTX A4000. Cet entrainement dure environ une minute l'évolution des fonctions de perte sur l'ensemble d'entrainement et celui de validation est représentée sur le graphique ci-dessous :

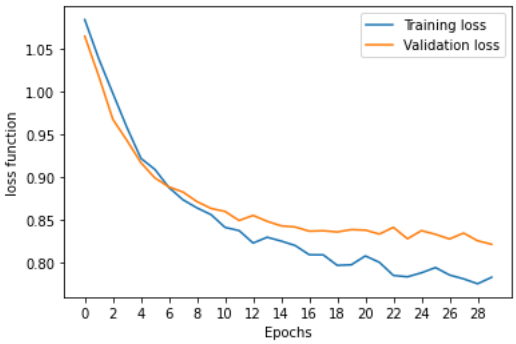


FIG. 10 – Evolution des fonctions de perte lors de l'entrainement du modèle FinBERT-ABSA

L'entraînement terminé, on teste notre modèle sur l'ensemble prévu à cet effet. On obtient finalement une accuracy de 0.74 plutôt satisfaisante dans le cadre d'une classification à 3 classes. Surtout, en analysant la matrice de confusion ci-dessous, on s'aperçoit que les erreurs du modèles consistent surtout à prédire un sentiment neutre lorsque le vrai label est positif ou négatif. Même si ces erreurs sont problématiques et doivent être améliorées, le fait d'attribuer à tort un score neutre à une entreprise semble moins "grave" que d'attribuer à tort un score positif ou négatif, notamment dans l'analyse de controverses.

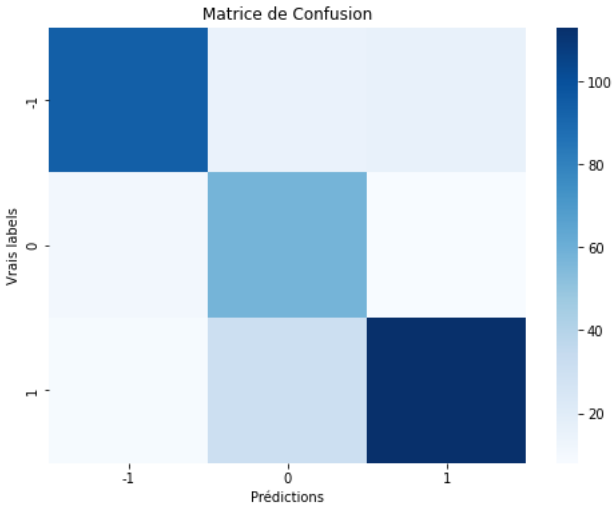


FIG. 11 – Matrice de confusion du modèle FinBERT-ABSA sur l'ensemble de test

L'entrainement et les performances de mon modèle semblent donc satisfaisants pour l'étude des news financières. En fine-tunant le modèle FinBERT sur seulement 1400 observations, le modèle est désormais capable de prendre en compte les informations ESG présentes dans le texte et de générer un sentiment spécifique à un aspect de la phrase dans près de 75% des cas. Je décide donc de sauvegarder les poids entrainés de ce modèle afin de l'appliquer, chaque jour, aux nouveaux flux d'informations disponibles.

5.3.5 Post-processing des données générées

Une fois cet entrainement effectué, le modèle est donc prêt à être utilisé sur les articles parus chaque jour. Ces news sont fournies sous forme de dataframe à un pipeline composé de l'ensemble des modèles et fonctions introduits précédemment afin de résoudre la tâche "E2E-ABSA" : Le modèle NER crée un nouveau dataframe contenant les entreprises extraites des news, ce dataframe est pré-processé afin d'être fourni au modèle et le score de sentiment est généré par le modèle FinBERT-ABSA. On parle ici de score de sentiment car la fonction softmax qui constitue l'output de mon modèle génère une probabilité d'appartenir à chaque classe. Pour une phrase donnée, le modèle ne va donc pas simplement fournir la classe pour laquelle la probabilité est la plus grande. J'ai choisis une telle représentation pour pouvoir être plus précis dans l'outil que je présente aux gérants de fonds. Une phrase pour laquelle le sentiment généré est "négatif" à 99% n'est effectivement pas vraiment similaire à une phrase négative à 51%.

A la sortie de ce modèle E2E-ABSA, on obtient donc un Dataset contenant, pour chaque titre de news, l'entité qui y est mentionnée et ses scores de sentiment associés. Ces informations sont contenues sous forme de tenseurs et un léger post-processing est nécessaire avant d'incorporer les autres modèles du pipeline et de présenter un outil d'aide à la décision.
Je commence par convertir ces données en dataframe. Ces dernières ne nécessitent plus de calculs importants et l'utiliation d'un dataframe est plus pratique pour la création de l'outil final.
En étudiant rapidement les résultats générés par le modèle NER, on s'aperçoit que certaines entreprises sont mentionnées plusieurs fois sous des noms qui diffèrent très légèrement. On retrouve par exemple dans les entreprises les noms "Apple" et "Apple's". Ces noms se réfèrent tout les deux à la même entreprise mais le modèle a extrait un "'s" en trop pour la deuxième entité. Ayant pour objectif de proposer un outil qui puisse filtrer les news et les sentiments pour chaque entreprise, ce dataframe nécessite un mapping afin de regrouper les diffèrents termes qui se réfèrent aux mêmes entreprises.
Afin d'effectuer ce mapping, j'ai décidé d'utiliser un mécanisme introduit brièvement en section 4.4.2 et

qui consiste à étudier les similarités entre les embeddings de chacune des entreprises. En mettant un seuil de similarité au dessus duquel on considère que les termes se réfèrent aux mêmes entreprises, cela peut permettre de regrouper les entreprises entre elles. J'ai donc utilisé un "sentence-transformers" afin d'obtenir les embeddings de chacune des entités extraites. Les sentence-transformers sont des modèles entrainés uniquement sur la représentation vectorielles des mots ou des phrases et qui sont réputés pour capturer au mieux les informations sémantiques. Une fois ces embeddings obtenus, j'applique une fonction qui calcule les similarités cosinus (2) entre chacun des ces vecteurs afin d'obtenir des scores de similarité entre les entités. En appliquant un seuil, je décide de regrouper entre elles toutes les entreprises dont la similarité cosinus $cos(\theta)$ et supérieure à 80%. En effectuant cette opération sur une base d'environ 1000 news, elle permet de modifier une soixantaine d'entreprises et les résultats semblent cohérents. J'ai finalement appliqué une fonction aux entités du dataframe afin de supprimer les symboles spéciaux qui peuvent appparaître lors de la détokenisation des outputs du modèle, la fonction *re.sub*. On obtient donc finalement un dataframe avec des données de bonne qualité et contenant les news, les entreprises et leurs scores de sentiment. Ce dataframe peut désormais être complété par les données que génèrent les autres modèles du pipeline avant d'être incorporé dans l'outil que nous avons créé.

5.4 Le pipeline final

Le dataframe que nous allons fournir chaque jour à l'outil d'aide à la décision doit être complété par les données suivantes pour chaque article: Le corps de l'article, un résumé de cet article et la traduction de ce résumé.
Je retrouve le corps de l'article à l'aide d'identifiants que je crée en recevant les données et avant de les fournir à mon modèle. Cet identifiant est nécessaire dans la mesure où les données sont mélangées aléatoirement lors de la création de batchs notamment. Il est stocké dans le dataframe (et dans les tenseurs pendant l'utilisation du modèle) et me permet d'effectuer un mapping entre la base de départ et celle générée par le modèle. Cela me permet de ne pas avoir à stocker le corps du texte pendant toutes les opérations du modèle, ce qui facilite réellement les calculs.

Une fois le corps du texte extrait, je décide d'utiliser le modèle BART présenté en section 4.4 afin de générer un résumé pour chaque article. Je charge pour cela les poids du modèle "bart-large-cnn" correspondant à la tâche de "summarization" à partir de la libraire HuggingFace ainsi que le tokenizer qui y est associé. Le modèle ne pouvant pas étudier de texte trop volumineux, j'extrais les 1000 premiers tokens pour effectuer le résumé dessus (en général les articles ne sont pas constitués de beaucoup plus de tokens). J'indique ensuite au modèle que le résumé doit contenir entre 30 et 130 tokens et je stocke tout les résumés générés par le modèle BART dans une liste que j'ajoute au dataframe initial.
Afin de générer des traductions pour chacun de ces résumés, j'utilise le modèle T5 présenté en section 4.5. Après avoir chargé le modèle "T5-base" de la librairie HuggingFace, j'effectue un léger pré-processing sur les résumés afin d'adapter le dataset à l'architecture "Texte-to-Texte" de T5. L'opération consiste simplement à ajouter le préfixe "translate English to French :" à chacun de ces résumés. Je stocke ces traductions dans une liste que j'ajoute également au dataframe.

Une fois ces deux dernières étapes effectuées, j'obtiens donc un dataframe comprenant pour chaque news, le corps de l'article, l'entreprise qui y est mentionnée, son score de sentiment, un résumé de la news et sa traduction. Ce pipeline composé des modèles NER, FinBERT-ABSA, BART et T5 peut donc être mis en production chaque jour sur les nouvelles news reçues par bloomberg.

Dans le but de fournir un outil aux gérants de fonds, nous avons finalement créé une application Dash afin d'obtenir une visualisation intéractive du dataframe fourni par mon pipeline sous forme de fichier CSV. Pour rendre cet outil efficient pour l'aide à la décision, nous avons décidé d'y ajouter la possibilité de filtrer toutes ces news en fonction de l'entreprise qui y est mentionnée mais également en fonction du score de sentiment calculé.
En filtrant uniquement les news associés à des scores de sentiment très négatifs, nous sommes par exemple en mesure de fournir un outil d'analyse de controverse vraiment intéressant dans le cadre d'une analyse ESG. Nous avons également décidé d'intégrer des graphiques à cette application afin d'obtenir, pour chaque entreprise, la répartition des news en fonction du sentiment calculé ainsi que l'évolution du score de sentiment moyen en fonction du temps.
Nous avons donc créé un outil d'aide à la décision basé sur de nombreux modèles de NLP afin de proposer aux gérants une analyse complète et adaptable de l'ensemble des entreprises constituant leurs portefeuilles. Des captures d'écrans de cet outil sont présentées en annexe.

6 Conclusion

En 1950, le scientifique Alan Turing prédisait que la machine et l'Homme pourraient converser naturellement au 21ème siècle et les derniers modèles publiés dans le domaine du NLP suscitent un sentiment d'optimisme par rapport à cette prédiction.
Depuis le début des années 2000 et l'introduction des réseaux de neurones, le domaine connaît effectivement un essor exceptionnel alimenté par des capacités de calcul et des bases de données toujours plus importantes. En représentant des données textuelles sous forme de vecteurs denses [10], Bengio et son équipe ont permis d'aborder l'analyse de langage et la génération de textes comme des problèmes de Machine Learning classiques. Ces embeddings de mots représentent en effet des données d'entrées parfaites pour les réseaux de neurones et permettent des calculs impossibles auparavant.

Mais c'est depuis six ans et l'introduction des modèles Transformers que le NLP se développe le plus rapidement. Cette architecture Transformers ainsi que les mécanismes qui y sont appliqués ont en effet permis de mettre en place des modèles capables de capturer les informations sémantiques et syntaxiques ainsi que des dépendances contextuelles complexes au sein de phrases ou de texte. Ces modèles disposent de capacités à comprendre et à générer le langage comparables à des capacités humaines. En utilisant cette architecture et en entraînant des modèles constitués de millions de paramètres tels que BERT ou T5 sur des bases de données constituées de milliards de mots, les chercheurs ont surtout permis l'application du NLP dans de nombreux domaines et avec des moyens limités. Ces modèles étant disponibles en Open-source sur des bases de données telles que HuggingFace, il est en effet possible d'en finetuner les poids (ou même l'architecture) en disposant de quantités de données négligeables par rapport à celles nécessaires pour un entraînement complet de tels modèles.

Ces modèles disponibles en open-source ont permis la réalisation de mon travail de recherche. J'ai décidé de me baser sur le modèle FinBERT, un modèle BERT adapté pour la compréhension de langage financier. En modifiant l'architecture du modèle et en le réentrainant pour extraire un sentiment par rapport à un aspect du texte étudié, j'ai finalement obtenu un modèle FinBERT-ABSA capable d'analyser les titres de news financières pour calculer un score de sentiment ESG par rapport à l'entreprise mentionnée dans le titre. En entraînant ce modèle sur un millier de news labellisées au préalables, j'obtiens finalement une accuracy de 73% qui paraît vraiment satisfaisante pour une classification à 3 classes et au vu de la faible quantité de données à ma disposition.
Les possibilités d'adaptation de modèles tels que BERT ou T5 permettent d'incorporer ce modèle au sein d'un pipeline afin de générer des traductions et des résumés pour étoffer l'outil d'aide à la décision et de nombreuses possibilités peuvent encore être envisagées pour améliorer cet outil. Les premiers retours nous ont d'ailleurs permis d'envisager d'autres possibilités de développement telles que la classification de news par thèmes mais également par secteurs d'entreprises.

7 Annexes

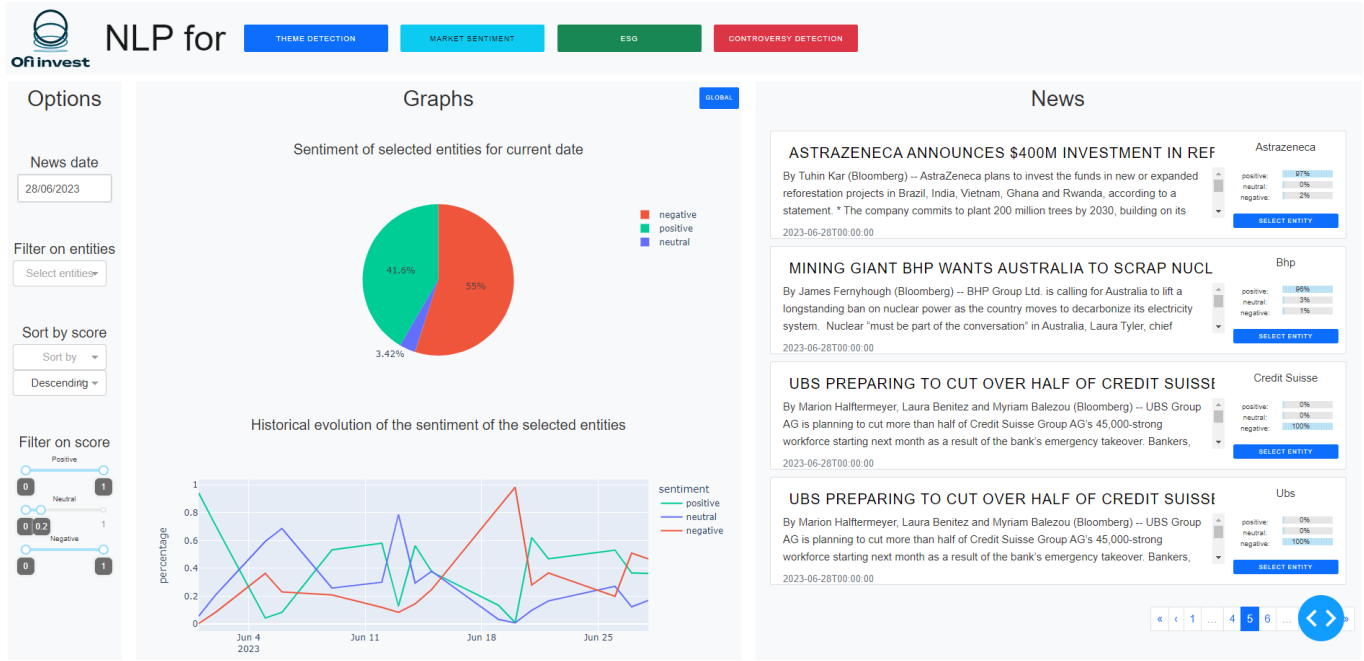


FIG. 12 – Visuel de l'outil d'aide à la décision

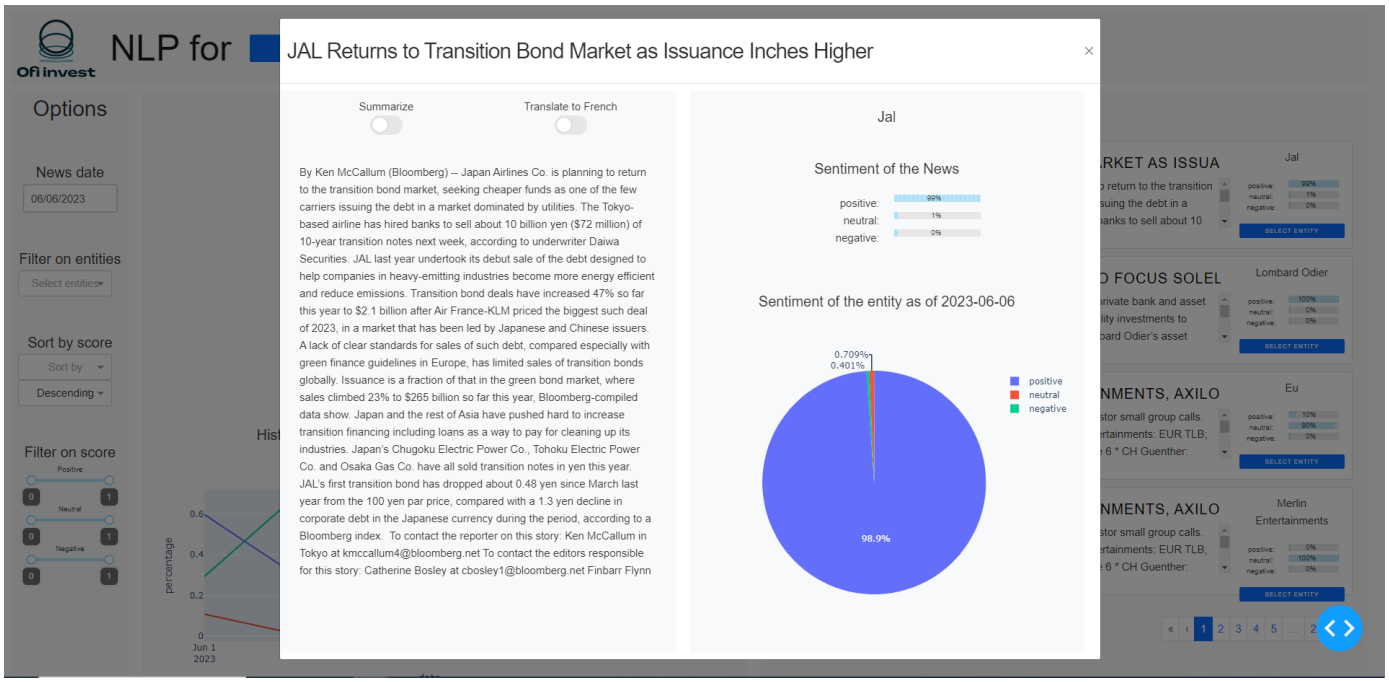


FIG. 13 – Visuel de l'analyse d'une news

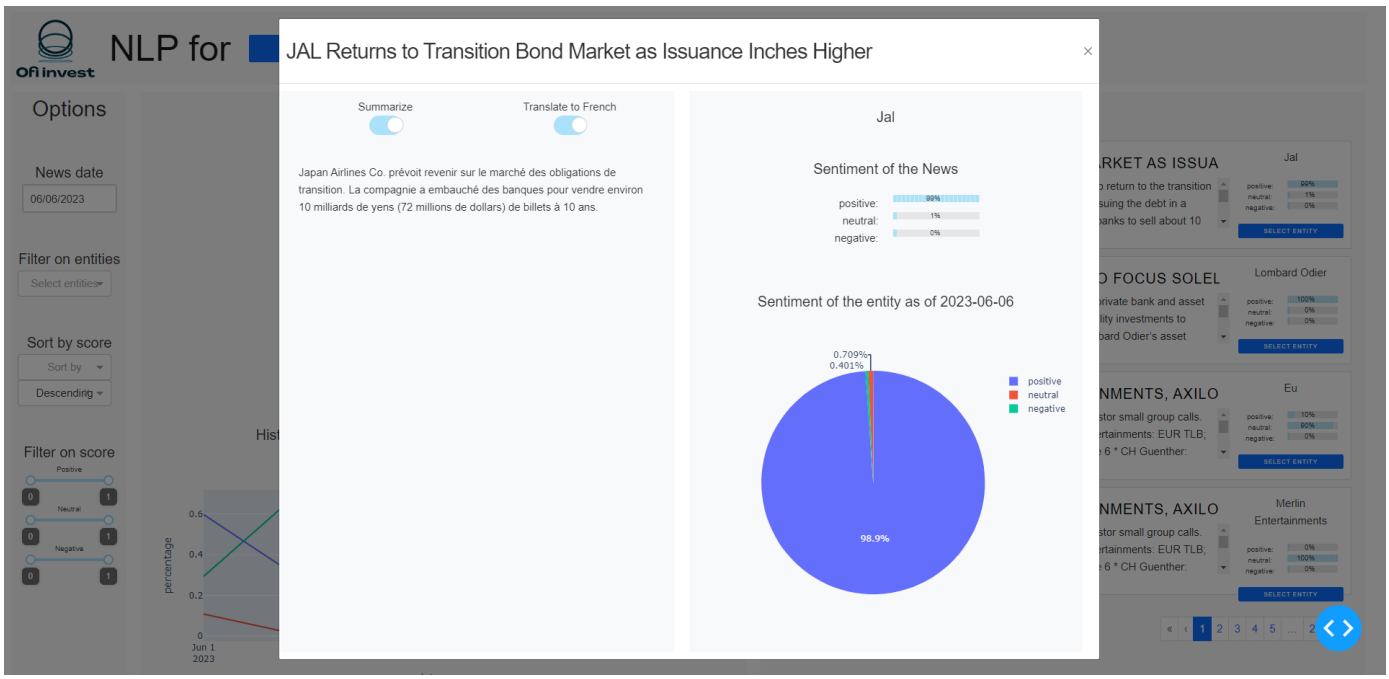


FIG. 14 – Visuel après summarization et translation de la news

Références

[1] Ashish Vaswani et al. “Attention Is All You Need”. In: (2017).

[2] Dzmitry Bahdanau et al. “Neural Machine Translation by jointly learning to Align and Translate”. In: (2014).

[3] Ilya Sutskever et al. “Sequence to Sequence Learning with Neural Networks”. In: (2014).

[4] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (2018).

[5] Mark Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: (2019).

[6] Schuster et al. “Japanese and Korean Voice Search”. In: (2012).

[7] Tomas Mikolov et al. “Efficient Estimation of Word Representations inVector Space”. In: (2013).

[8] Tomas Mikolov et al. “Reccurent neural network based language model”. In: (2010).

[9] Wenxuan Zhang et al. “A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges”. In: (2022).

[10] Yoshua Bengio et al. “A Neural Probabilistic Language Model”. In: (2003).

[11] Zihan Liu et al. “NER-BERT: A Pre-trained Model for Low-Resource Entity Tagging”. In: (2021).

[12] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: (2020).

[13] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: (2019).

[14] Dogu Tan Araci. “FinBERT: Financial Sentiment Analysis with Pre-trained Language Models”. In: (2019).

[15] Noam Chomsky. Syntactic Structures. Mouton Co, 1957.

[16] Diederik P. Kingma et Jimmy Lei Ba. “ADAM: A method for stochastic optimization”. In: (2014).

[17] Laurens van der Maaten et Geoffrey Hinton. “Visualizing Data using t-SNE”. In: (2008).

[18] René Arnulfo et Yulia Ledeneva. “Word Sequence Models for Single Text Summarization”. In: (2009).