: N° d'ordre
: N° de série

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude
Présenté pour l'obtention du Diplôme de

MASTER ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes Distribués et Intelligence Artificielle**

Présenté par :

- **Otmani Abdelfattah**
- **Bouras Abdelali**

Thème

Architecture de transformateur pour la modélisation de la langue arabe.

Soutenue le 22-09- 2022 Devant le jury:

M.	Lajdal Ebrahim	MCA	Président
M.	Bali Ahmad	MAA	Rapporteur
M.	Naoui Mohammed Anouar	MAA	Encadreur

Année Universitaire: 2021/2022

Résumé

L'arabe est une langue morphologiquement riche avec relativement peu de ressources et une structure moins explorée par rapport à l'anglais.En raison de ces limitations, les tâches de traitement de la langue arabe naturelle (TAL) .

Récemment, les modèles basés sur le langage tels que AraBERT et BERT se sont avérés très efficaces pour comprendre le langage, à condition qu'ils soient pré-formés sur un ensemble de données. Ces modèles ont pu établir de nouvelles normes et obtenir de meilleurs résultats pour la plupart des tâches de la TAL. Dans notre travail nous avons analysé les sentiments par l'utilisation de transformateur AraBERT.

Mots clés : traitement de la langue arabe naturelle (TAL) ,AraBERT,BERT.

Abstract

Arabic is a morphologically rich language with relatively few resources and a less explored structure compared to English. Because of these limitations, Natural Arabic Language Processing (NLP) .

Recently language-based models such as AraBERT and BERT have proven to be very effective in understanding language, provided that they are pre-trained on a set of data. These models were able to set new standards and achieve improved results for most NLP tasks. In this work we analyzed sentiment AraBERT transformer.

Keywords : Natural Language Processing (NLP),AraBERT,BERT

الملخص

اللغة العربية هي لغة غنية شكليًا مع موارد قليلة نسبيًا وبنية أقل استكشافًا مقارنة باللغة الإنجليزية. بسبب هذه القيود ، ثبت أن مهام معالجة اللغة العربية الطبيعية (NLP) صعبة للغاية في التعامل معها

في الآونة الأخيرة أثبتت النماذج المعتمدة على لغة مثل AraBERT و BERT أنها فعالة جدًا في فهم اللغة ، بشرط أن يتم تدريبهم مسبقًا على مجموعة من البيانات . كانت هذه النماذج قادرة على وضع معايير جديدة وتحقيق نتائج محسنة لمعظم مهام البرمجة اللغوية العصبية. في هذه المذكرة، عملنا على تحليل المشاعر من خلال التصنيف في اختار مدا قوة AraBERT

كلمات مفتاحية :

معالجة اللغة الطبعية ، AraBERT ، BERT

Table des matières

Résumé . . . . . **I**

Abstract . . . . . **II**

III . . . . . **الملخص**

Introduction générale **1**

1 Traitement automatique des langues **2**
1.1 Introduction : 3
1.2 Bref historique du TAL : 3
1.3 Les niveaux de traitement en TAL : 4
1.3.1 Niveau morpho lexical (morphologique) : 4
1.3.2 Niveau syntaxique : 5
1.3.3 Niveau sémantique : 5
1.3.4 Niveau pragmatique : 5
1.4 Les problèmes majeurs de TAL : 5
1.4.1 L'Ambiguïté : 5
1.4.2 L'Implicite : 6
1.5 Architecture d'un système TAL : 6
1.5.1 Architecture startificationnelle (Appelée aussi séquentielle ou en série) : 6
1.5.2 Architecture moins hiérarchisée (Appelée aussi systèmes hétérarchiques,parallèles ou intégrés) : 7
1.6 Les applications de TAL : 8
1.6.1 Le traitement documentaire : 8
1.6.2 La production de documents : 10
1.6.3 Les interfaces naturelles : 11
1.7 Conclusion : 12

2 Deep Learning et réseaux neurones récurrents **13**
2.1 Introduction 14
2.2 Définition de l'apprentissage profond (deep learning) : 14
2.3 réseaux de neurons : 15
2.3.1 Concepts de base 15
2.3.2 Neurone formel 15
2.4 Réseaux de neurones récurrents (RNN) 19
2.4.1 Long Short-Term Memory (LSTM) 21

2.4.2 Long Short-Term Memory Bidirectionnels (BLSTM) 23
2.5 Conclusion 25

3 Le modèle de transformateur pour l'analyse des sentiments 26
3.1 Introduction 27
3.2 Le Transformateur 27
3.3 L'Attention 28
3.4 BERT 29
3.5 Les étapes de notre travail 32
3.6 Outils et Environnement de programmation 35
3.6.1 Environnement matériel 35
3.6.2 Environnement logiciel 35
3.7 Conclusion 42

Conclusion et perspectives 43

Bibliographie 45

Table des figures

1.1 Les niveaux de TAL 4
1.2 Architecture startificationnelle d'un système TALN[30] 7
1.3 Architecture startificationnelle d'un système TALN[30] 7

2.1 Différences entre deux approches de classification de la polarité des sentiments, l'apprentissage automatique (en haut) et l'apprentissage en profondeur (en bas). Partie du discours (POS) ; Reconnaissance d'entité nommée (NER) ;Terme Fréquence-Inverse Document Frequency (TF-IDF).[4] 15
2.2 Un perceptron multicouches (MLP) à une seule couche cachée. Les neurones de la couche d'entrée sont représentés par des demi-cercles car ils ne déterminent pas des potentiels postsynaptiques biaisés. 16
2.3 Représentation compacte des RNN. Toutes les flèches représentent des connexions complètes. La flèche en pointillée représente les connexions ayant un décalage temporel ($t-1$). 19
2.4 Représentation dépliée des RNN 20
2.5 Représentation dépliée hiérarchisée des RNN 21
2.6 Une cellule Long Short-Term Memory (LSTM). Les flèches en pointillé représentent les opérandes avec un décalage temporel ($t-1$). La fonction d'activation α (pour l'entrée et la sortie) est généralement la tangente hyperbolique tanh. 22
2.7 Un RNN bidirectionnel (Bidirectional RNN ou BRNN). 23

3.1 Le Transformateur - modèle d'architecture [35] 28
3.2 Le Transformateur - modèle d'architecture [35] 29
3.3 Procédures globales de pré-entraînement et d'ajustement fin pour BERT. A part les couches de sortie, les mêmes architectures sont utilisées pour le pré-entraînement et le réglage fin. Les mêmes paramètres de modèle pré-entraînés sont utilisés pour initialiser les modèles pour différentes tâches en aval. modèles pour différentes tâches en aval. Lors de l'ajustement fin, tous les paramètres sont ajustés. [CLS] est un symbole spécial spécial ajouté devant chaque exemple d'entrée, et [SEP] est un symbole spécial de séparation (par exemple, pour séparer les questions des réponses).[9] 30
3.4 Les étapes de notre modèle 32
3.5 architecture du réglage fin 32
3.6 architecture AraBERT dans 'analyse des sentiments 33
3.7 Logo AraBERT 33
3.8 Répertorier l'ensemble de données 34
3.9 model AraBERT 35

3.10 interface Anaconda 36
3.11 Logo PyTorch 37
3.12 Télécharger modele 37
3.13 Vérifiez la longueur de la séquence du modèle 38
3.14 Longueurs des phrases d'entraînement 38
3.15 Longueurs des phrases a tester 38
3.16 Partie du code montrant le processus de classification des datasets. 39
3.17 interface graphique De l'application 39
3.18 Rapport de classification 40
3.19 tableau de confusion 41
3.20 Évolution de la précision et du taux d'erreur AraBERT 42

Liste des tableaux

1.2 Bref historique du TAL [12] 3

3.2 Les résultats des valeurs statistiques du processus de mise en œuvre du modèle 40

Introduction générale

Internet est devenu un outil indispensable dans le travail et dans la vie quotidienne, notamment avec la croissance massive et rapide de réseaux sociaux.

La popularité des médias sociaux est liée à la demande d'information qui devient de plus en plus importante dans notre société. Généralement, les gens expriment ce qui est en eux à travers les médias sociaux et aiment aussi voir les réactions et les interactions des autres par exemple avant de prendre une décision.Cependant, le grand volume d'informations générées sur ces médias sociaux nécessite des outils suffisants pour les traiter et les analyser.

L'analyse des sentiments est un domaine de la TALN ayant pour but d'analyser le ressenti d'un utilisateur sur les réseaux sociaux à travers ses retours pour prendre une décision. Prise de décision dans divers domaines : politique, marketing, santé, éducation, etc.

L'analyse des sentiments peut améliorer les capacités des systèmes de gestion de la relation client et de recommandation, par exemple en aidant à découvrir les caractéristiques d'intérêt particulier pour les clients ou en excluant les éléments qui reçoivent des critiques défavorables des publicités.Le problème d'analyse des sentiments exprimé dans la méthode d'analyse des sentiments permet de classer le texte comme positif, négatif. La classification des sentiments des textes arabes n'a pas reçu la même attention que l'anglais, en raison premièrement du manque d'outils et de ressources pour extraire les sentiments arabes des textes en arabe, et deuxièmement du caractère unique de l'arabe, car il est spécial, donc les mots peuvent avoir de nombreux sens. Et la composition est riche, peut complètement changer la question, et un mot peut renvoyer à plusieurs sens différents selon la phrase. Il peut s'agir du nom d'une personne ou d'un adjectif, ou il peut être formé en insérant des matériaux auxiliaires (suffixes) pour former diverses formes ou préfixes.

Dans notre travail, nous visons à explorer le domaine de l'analyse des sentiments dans les textes arabes et à les classer avec des émotions positives ou négatives en utilisant des techniques d'apprentissage en profondeur utilisant l'adaptateur, et nous avons utilisé le modèle AraBERT. Et ça nous a donné de bons résultats.

Notre travail est organisé comme suit : Dans le premier chapitre, nous donnons une définition idiomatique du traitement du langage naturel. Dans le deuxième chapitre, nous avons traité des réseaux de neurones récursifs. Au chapitre trois, nous avons fait le transformateur, AraBERT, la conception et à la mise en œuvre de notre travail.

Chapitre 1

Traitement automatique des langues

1.1 Introduction :

La Science du traitement automatique des langues naturelles (TALN), comme des langues parlées, lues et écrites par les êtres humains, et l'ensemble des langages informatique, artificiels, mathématiques ou logiques, nous permet de produire un système linguistique facile et intelligible.[34]
D'une manière plus précise, le TALN est la conception de logiciels capables de traiter de façon automatique des données exprimées dans une langue (dite « naturelle », par opposition aux langages formels de la logique mathématique). Il se constitue de deux grands domaines d'étude qui partagent un seul objectif : le traitement d'une manière automatique des langues naturelles via l'utilisation des programmes et logiciels informatiques pour traiter des données (des corpus, des documents, des textes, des phrases, des mots, etc.) à la base des règles et grammaires linguistiques. Autrement dit, pour faire une certaine application du TALN, il faut déterminer ou créer un programme, un logiciel approprié à cette application et en parallèle de déterminer les règles grammaticales associées à l'application traitée. Parmi les applications les plus connues du TALN, nous pouvons citer : la traduction automatique, la correction orthographique, la recherche d'information et la fouille de textes, le résumé automatique, la génération automatique de textes, la synthèse de la parole, la reconnaissance vocale, la reconnaissance de l'écriture manuscrite, etc.

Dans ce chapitre, nous présentons le traitement automatique des langues, leur niveaux de traitement et les problèmes , l'architecture d'un système de TAL, et les applications du TAL.

1.2 Bref historique du TAL :

L'année	**Le traitement**
Années 50	Traduction automatique – débuts du TAL
1964	Rapport ALPAC
Années 60	. Linguistique formelle (Chomsky, Montague) comme base pour le TAL . Applications basées sur des techniques linguistiques (Eliza, shrdlu) – Chomsky (grammaires formelles, analyseurs syntaxiques) ; sémantique procédural (Woods) . Approches limitées à des domaines restreints. Non portables.
Années 70	Premières applications
Années 80	. Approches symboliques. Applications utilisent des connaissances linguistiques et encyclopédiques extensives. Manquent de robustesse.
Années 90 et plus	Premiers corpus, approches statistiques, apprentissage automatique. Applications utilisent corpus de grande taille et méthodes statistiques.

Tab. 1.2 : Bref historique du TAL [12]

1.3 Les niveaux de traitement en TAL :

Dans cette partie, nous passerons en revue les différents niveaux de traitement requis pour une compréhension complète de l'énoncé du langage naturel. Chacun de ces niveaux implique un traitement très spécial, permettant de réaliser le concept du système standard en traitement automatique.

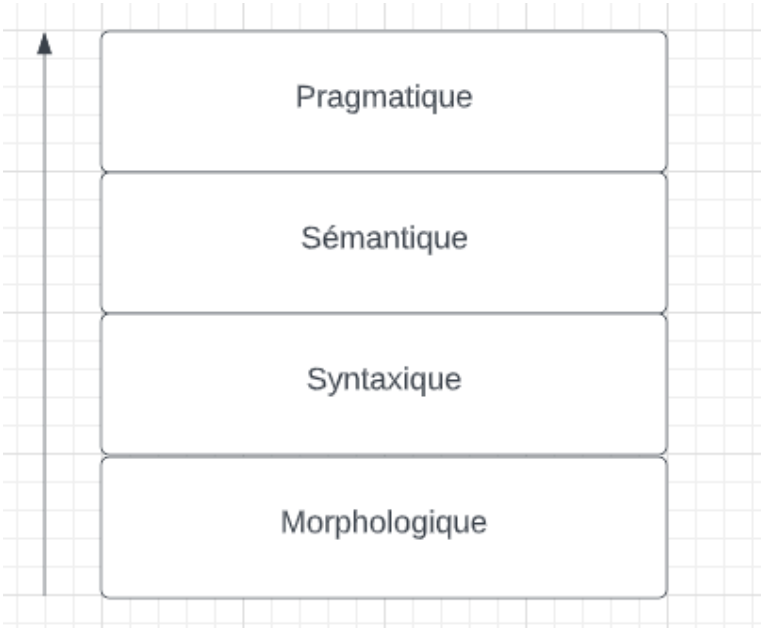

Fig. 1.1 : Les niveaux de TAL

1.3.1 Niveau morpho lexical (morphologique) :

À ce niveau, nous étudions la composition des mots et le programme de diversité en Formes. Deux termes doivent être étudiés : la flexion et la dérivation :

La flexion :

C'est un ensemble de modifications que subit un mot à sa fin. par rôle Il se joue dans des phrases, et l'inflexion est un processus qui consiste à modifier les radicaux des mots pour ajouter certains types d'éléments.[37]

La dérivation :

C'est le processus d'ajout, de suppression ou de formation de nouveaux mots remplacer les éléments grammaticaux d'un mot simple.[37]

1.3.2 Niveau syntaxique :

À ce niveau, nous soucions de l'ordre des mots et de leurs relations structurelles. Celui-là Les connaissances grammaticales se rapportent à la façon dont les mots sont arrangés en phrases, C'est-à-dire sa structure grammaticale. Ces connaissances ou règles sont décrites en grammaire et leur application permet la formation de phrases correctes et la clarification.

1.3.3 Niveau sémantique :

La sémantique est une discipline conçue pour décrire le sens Spécifique au langage et à son organisation théorique. En TAL, la sémantique peut être Défini comme l'étude du sens des mots, des phrases et des déclarations.
Par conséquent, le rôle de l'analyseur sémantique est d'attribuer un sens aux phrases structurées. par l'analyseur.
Les connaissances sémantiques requises pour comprendre le nom sont distinctes Il explique non seulement la relation entre les mots et leurs objets, actions ou idées désignés, mais aussi les conditions dans lesquelles une phrase est jugée significative.

1.3.4 Niveau pragmatique :

Pour bien comprendre le livre, il faut aussi posséder des connaissances pratiques, c'est-à-dire des informations qui permettent de replacer le mot dans le contexte de la connaissance pragmatique, c'est-à-dire des connaissances qui permettent de replacer le mot dans son contexte. La connaissance pragmatique fournit une représentation du monde de référence qui agit comme une culture commune pour les interlocuteurs. Étant donné que de nombreuses allégations ne peuvent être comprises que dans un contexte géographique, historique ou culturel particulier, le niveau pratique est le plus difficile à comprendre pour la machine.

1.4 Les problèmes majeurs de TAL :

Les problèmes rencontrés en TAL sont principalement de deux types : l'ambiguïté du langage et la quantité d'informations implicites incluses dans les communications naturelles. interaction naturelle

1.4.1 L'Ambiguïté :

À tous les niveaux, le langage naturel n'est pas clair. Cette ambiguïté, loin d'être accidentelle, est l'un de ses traits distinctifs. Elle peut être considérée comme une conséquence inévitable d'un compromis entre une capacité d'expression quasi illimitée d'une part, et les contraintes imposées par les ressources physiologiques limitées disponibles d'une part. (taille de la mémoire à long terme et à courte portée, densité de l'espace acoustique, contraintes de charnière, etc).[40]

Comme le montrent les exemples suivants, il existe plusieurs interprétations alternatives pour chacune des entités linguistiques importantes pour un niveau de traitement :

- ambiguïté du graphème (lettre) dans le processus d'encodage orthographique : comparez le son du I dans lit, poire et maison
- ambiguïté dans les propriétés grammaticales et sémantiques (c'est-à-dire celles liées à son sens) d'une forme graphique donnée : par exemple, manges est ambiguë à la fois sur le plan morpho-syntaxique (parce qu'elle correspond aux formes indicative et subjonctive du verbe manger) et sur le plan sémantique (parce qu'elle correspond aux formes indicative et subjonctive du verbe manger).
- l'objectif grammatical des groupes de mots n'est pas clair.
- la portée des quantificateurs, des conjonctions et des prépositions n'est pas claire.
- l'incertitude quant à la façon dont une remarque doit être interprétée dans son contexte.[40]

1.4.2 L'Implicite :

Le langage fait toujours partie de l'interaction entre deux personnes qui sont censées être équipées d'une telle connaissance du monde et de son fonctionnement, de sorte que la grande majorité des facteurs contextuels nécessaires pour clarifier l'ambiguïté, ainsi que la compréhension, la parole naturelle peuvent rester implicites. Lorsque la machine tente d'entrer dans un processus de communication naturelle avec un être humain, la situation change complètement : la machine n'a pas cette connaissance de base, ce qui rend difficile, voire impossible, de comprendre pleinement la majorité des phrases sans règles de connaissance supplémentaires, qui donnent accès à une connaissance constante et dynamique du monde (ou du domaine) en général (connaissance statique) (connaissance dynamique).[40]

1.5 Architecture d'un système TAL :

En général, les systèmes TAL peuvent être divisés en deux éléments d'architecture différents :[30]

1.5.1 Architecture startificationnelle (Appelée aussi séquentielle ou en série) :

Dans ce système, nous avons chaque type de connaissance, nous avons une unité informatique différente que nous lui transmettons les données fournies par l'unité précédente en entrée, sans aucune observation possible. Le graphique suivant montre un exemple de structure initiale Un système conçu pour extraire des représentations pratiques du texte écrit :

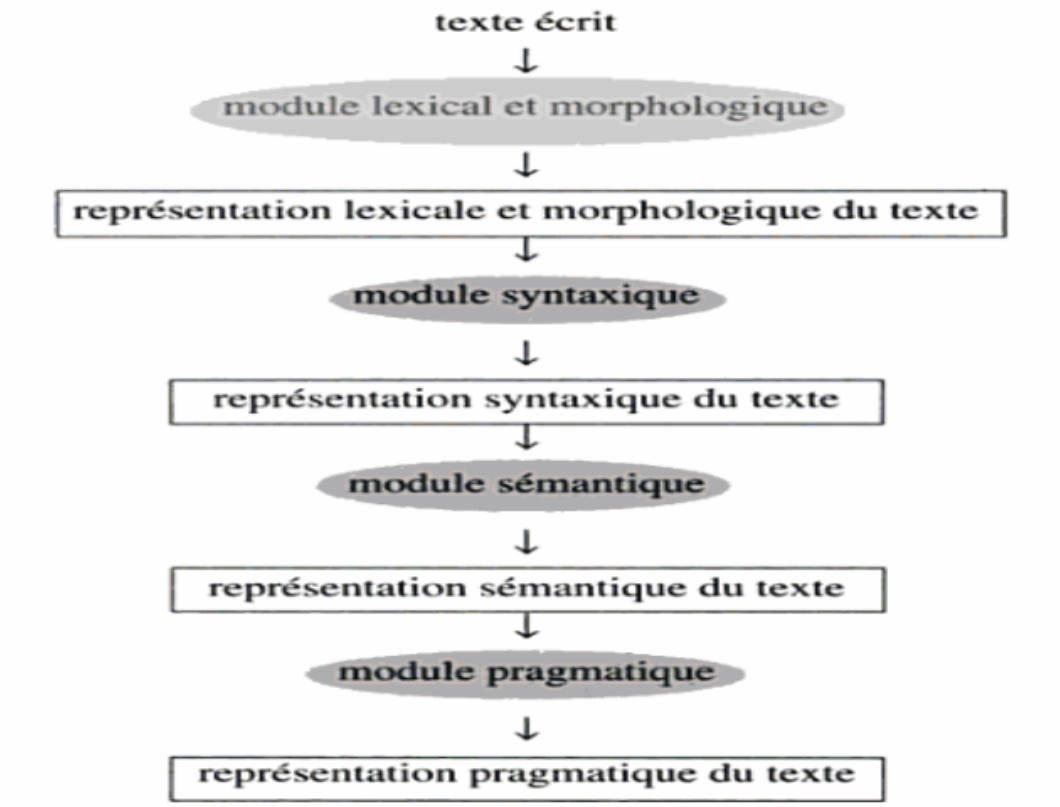

FIG. 1.2 : Architecture startificationnelle d'un système TALN[30]

1.5.2 Architecture moins hiérarchisée (Appelée aussi systèmes hétérarchiques,parallèles ou intégrés) :

Ce type d'architecture utilise simultanément différentes connaissances, soit en intégrant différentes connaissances au niveau du mot dans des représentations complexes, soit en faisant communiquer différents modules. Les avantages de ces architectures sont évidents en intégrant différentes Les connaissances, qui évitent l'ambiguïté artificielle, sont propres aux machines, et celles qui surgissent lorsque les connaissances sont considérées les unes après les autres :

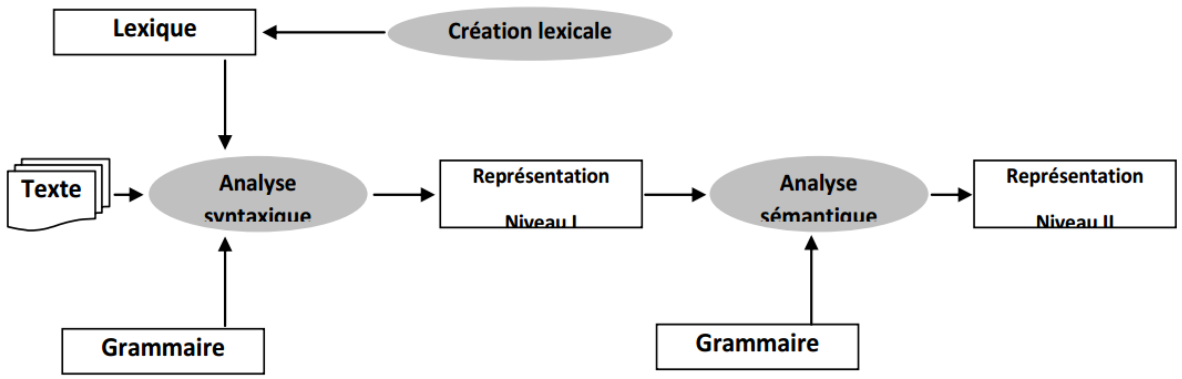

FIG. 1.3 : Architecture startificationnelle d'un système TALN[30]

1.6 Les applications de TAL :

En termes d'applications, le besoin en TAL découle de deux sources, pour faire court. D'une part, afin de résister à leur développement exponentiel, il est nécessaire de pouvoir traiter (créer, lire, rechercher, catégoriser, analyser et traduire) les informations textuelles de manière de plus en plus "intelligente". Par conséquent, les approches du TAL ont un large éventail d'applications. Ces applications ont été divisées en trois catégories : les aides à la lecture de documents, les aides à la création de documents, et enfin les interfaces homme-machine.[40]

1.6.1 Le traitement documentaire :

Les applications les plus immédiates de la PNL sont celles qui tentent d'aider le traitement humain des vastes ressources de langage naturel accessibles, comme par exemple :

La traduction automatique :

Cette application, qui a suscité les premiers efforts de recherche en matière de TAL, reste une préoccupation économique et politique essentielle. Si de tels interprètes existaient, il serait beaucoup moins nécessaire de s'appuyer sur une langue "universelle" comme l'anglais pour assurer la distribution mondiale des documents.Il convient de noter que l'on est encore loin d'une traduction complète indépendante du domaine,Il est possible de trouver des traducteurs experts compétents (dans des domaines techniques), ce qui constitue une excellente technique pour se préparer à l'intervention manuelle d'un traducteur. Il existe des écosystèmes supplémentaires qui fournissent des ressources lexicales (grands dictionnaires bilingues) pour l'aide à la traduction. Enfin, il existe une utilisation intrigante de systèmes de traduction entièrement automatiques, fondée sur l'idée que la recherche de documents, ou le filtrage manuel de documents, qui peut impliquer la lecture en diagonale d'un très grand nombre de documents, est toujours mieux effectuée dans la langue maternelle de l'utilisateur...[12]

Aujourd'hui, plusieurs solutions commerciales (Systran, Logos, Metal, Aleth-Trad, etc.) sont disponibles, et certains moteurs de recherche proposent la traduction automatique des pages html.[12]

La recherche documentaire :

Dans les bases de données documentaires, recherchez des documents « intéressants ». La croissance des moteurs de recherche de documents Web, qui traitent des millions de requêtes chaque jour, démontre l'importance des besoins de ce domaine. Les performances de ces moteurs montrent jusqu'où ils vont. Nous devons encore aller dans ce domaine. D'autres moteurs de recherche méritent d'être bien connus, même si Google semble s'imposer aujourd'hui.[12]

De plus en plus de solutions offrent des fonctionnalités telles que le suivi automatique

des publications dans certaines zones ou la recherche automatique d'adresses potentiellement pertinentes (en fonction des profils d'utilisateurs).[12]

Le Résumé automatique :

Le résumé automatique est le processus de réduction de la matière d'un texte source en un résumé par la sélection et/ou la généralisation de ce qui est significatif dans le texte source.[37]

Le résumé automatique de texte semble être une solution viable à cheval sur deux disciplines : traitement automatique de la langue (TAL) et les données de recherche (RI). Le résumé automatique de texte crée une représentation d'un texte qui est plus courte que l'original tout en conservant les informations importantes.[10]

La plupart des universitaires se sont appuyés sur des méthodes basées sur la connaissance linguistique pour construire des systèmes automatiques de résumé de texte. Ces systèmes reposent sur des techniques d'extraction, qui partent du principe qu'il faut faire ressortir les informations pertinentes en choisissant les phrases qui les définissent.[10]
Les stratégies d'extraction sont basées sur :

- La relation entre les termes du titre du texte et leur apparition dans le texte source.
- Un examen thématique et structurel du discours.
- La création de relations de cohésion lexicale entre les phrases afin d'extraire celles qui sont les plus liées.
- L'emploi de marqueurs rhétoriques tels que justification, cause et justification, cause, consécution, contraste, conséquence, etc...[10]

La Reconnaissance de caractères :

Voici le principe général d'un système de reconnaissance optique de caractères :

1. Scanner pour convertir les textes en images.
2. Utilisation de techniques de reconnaissance de formes (lettres) en conjonction avec l'apprentissage (réseaux neuronaux, HMM)
3. Utilisation d'un modèle de langage (contenant des ressources telles que des dictionnaires et des grammaires) pour obtenir l'hypothèse la plus probable.

Applications pratiques : Dématérialisation de documents (bibliothèque), dématérialisation de formulaires (contrôles, administration), adresses de tri postal et identification d'inscription.
Industriels du domaine :Nuance (ScanSoft / Xerox), ABBYY, IRIS, Novo Dynamics,Datacap, EDT, Ligature.[8]

La Correction orthographique / grammaticale :

Voici les principales fonctions d'un correcteur orthographique :

- Reconnaissance des mots (tokenisation).
- Vérifier l'orthographe : les mots qui ne sont pas dans le dictionnaire et qui ne sont pas dans une langue étrangère, ainsi que les noms propres, les chiffres et les acronymes...
- Correction grammaticale : détermine le rôle du mot dans la phrase (spécifié, nom, verbe, circonstance, etc.), puis nous le travaillons dans des règles grammaticales pour l'analyse grammaticale.

Applications pratiques : correction de document rédigé par des étudiants dont le niveau de français est généralement bas.
Industriels du domaine : Synapse, Druide, Microsoft, Diagonal.[8]

D'autres applications sont issues du traitement documentaire, comme par exemple :

- Les applications du paradigme de la recherche de documents comprennent le routage, la catégorisation et l'indexation automatique des documents électroniques.
- rouver (ou générer à la demande) des réponses exactes aux demandes de l'utilisateur est un défi plus difficile (tâche "question-réponse").
- Lecture automatisée de fichiers, par exemple, pour stocker des fichiers dans des structures de données officielles ou pour en extraire des résumés.
- Analyse d'un corpus de textes sur un certain sujet (histoire, stylométrie, veille technologique, etc.). La mise à disposition d'outils de visualisation et d'exploration interactive de sujets disciplinaires est une utilisation courante dans ce secteur (par exemple dans le domaine scientifique).[40]

1.6.2 La production de documents :

Parce que quelqu'un a écrit tant d'articles électroniques, ils sont maintenant largement disponibles. Les applications du TALN abondent dans le domaine de l'aide à la création de textes (génération de textes) :

* claviers "autocorrectifs" (pour les personnes handicapées, par exemple) ;
* reconnaissance optique de caractères De nombreuses solutions commerciales sont désormais disponibles, avec des performances extrêmement excellentes : Recognita, Omnipage, ScanWorX, etc ;

* La production de documents à partir d'exigences formelles est automatisée. En fait, un grand nombre d'industries reposent sur la fabrication en masse de textes fortement stéréotypés basés sur des normes plus ou moins formelles (textes juridiques, rapports d'exploration de bases de données rapports d'exploration de bases de données, rapports d'analyse statistique, documentation technique, etc.)). (Par exemple, les documents techniques). Il est absolument envisageable de créer automatiquement, pour ce type de documents, sinon des textes totalement définitifs, du moins des versions préliminaires qui seront finalisées par des rédacteurs humains.

On peut retrouver dans ces systèmes la même dialectique que dans les applications de gestion documentaire. D'une part, il existe des applications à couverture étendue qui utilisent principalement des ressources lexicales et incluent des fonctionnalités de tolérance d'accès au lexique (permettant la réparation des erreurs) au lexique : par exemple, les applications de correction orthographique. D'autre part, les programmes qui combinent des techniques de traitement (généralement la génération) mais qui ne sont applicables qu'à un ensemble limité de domaines. ne fonctionne que pour un ensemble de domaines considérablement réduit.[40]

1.6.3 Les interfaces naturelles :

Le domaine des interfaces naturelles (c'est-à-dire en langage naturel), telles que :

- Moteurs de recherche Web ou interrogation de bases de données en langage naturel (traduction en langage naturel SQL). Ce type d'application est de plus en plus courant sur l'internet.

- Les interfaces vocales, qui mettent en œuvre la reconnaissance et la synthèse vocales, la production et la gestion de dialogues, l'accès à des bases de connaissances, etc. de diverses manières selon les applications, chacun de ces modules nécessitant des traitements particuliers (désambiguïsation morpho-syntaxique et identification de syntagmes pour la synthèse, grammaire stochastique pour la reconnaissance vocale...). Chacun de ces composants nécessite un traitement particulier (désambiguïsation morpho-syntaxique et identification de syntagmes pour la synthèse, grammaires stochastiques pour la reconnaissance de la parole...). Les premiers systèmes de dictée vocale "grand public" commencent à apparaître sur le marché (Via Voice, l'offre d'IBM, Dragon Dictate, et bien d'autres), et l'intégration d'une API de traitement de la parole dans Windows devrait faire exploser le marché des technologies vocales dans les prochaines années. Les premiers systèmes de dictée vocale "grand public" commencent à apparaître sur le marché (Via Voice, l'offre d'IBM, Dragon Dictate, et bien d'autres), et l'incorporation d'une API de traitement de la parole dans Windows devrait faire exploser le marché de la technologie vocale dans les années à venir. De nombreux services commerciaux utilisant l'ensemble de ces techniques (reconnaissance-dialogue-synthèse) pour diverses applications telles que les ordinateurs "de poche", les ordinateurs "mains libres", la lecture téléphonique du courrier électronique, la réservation de billets (train), etc. existent déjà ou sont proches de la commercialisation. La liste pourrait encore s'allonger. En 1995, le marché de la

technologie vocale aux Etats-Unis valait quelques centaines de millions de dollars, et il vaut aujourd'hui plusieurs milliards de dollars.En 1995, le marché de la technologie vocale valait quelques centaines de millions de dollars en dollars américains, mais en 2000, on prévoyait qu'il vaudrait environ 3 milliards de dollars.[40]

1.7 Conclusion :

L'étude des langages et des mécanismes permettant de réaliser des traitements automatiques par des machines est un domaine d'étude foisonnant et riche d'applications potentielles ou émergentes. Il reste encore beaucoup de progrès à faire pour mieux comprendre cette faculté et construire des systèmes capables de soutenir la comparaison avec l'homme, Mais l'état actuel des connaissances nous permet d'offrir de nombreuses solutions efficaces aux problèmes et exigences réels.

A travers ce premier chapitre, nous avons mis le point sur les concepts de base de TAL, nous avons ainsi exploré les principales générations des produits et systèmes TAL.

Et le semestre prochain, nous en apprendrons davantage sur l'apprentissage profond et les réseaux de neurones répétitifs, et les utiliserons dans Le Traitements Automatique des Langues (TAL)

Chapitre 2

Deep Learning et réseaux neurones récurrents

2.1 Introduction

Ces dernières années, les performances particulières des réseaux de neurones récurrents (RNN) dans le traitement séquentiel des données sont devenues à la pointe de la technologie dans divers domaines d'application. Les architectures de type mémoire à long terme (LSTM) [16] sont notamment basées sur les RNN, elles sont plus efficaces et réduisent la perte d'informations dans le cas de séquences longues. Ces paires de structures comprennent des séquences de mots [33], des séries temporelles [13] , Image [36]. Dans ce chapitre, nous citons les avantages sur l'apprentissage en profondeur et les réseaux de neurones récursifs et leurs techniques

2.2 Définition de l'apprentissage profond (deep learning) :

L'apprentissage profond adapte une approche multicouche aux couches cachées du réseau neuronal. Dans les approches traditionnelles d'apprentissage automatique, les caractéristiques sont définies et extraites soit manuellement, soit en utilisant des méthodes de sélection des caractéristiques. Cependant, dans les modèles d'apprentissage profond, les caractéristiques sont apprises et extraites automatiquement, ce qui permet d'obtenir une meilleure précision et de meilleures performances. En général, les hyper paramètres des modèles de classification sont également mesurés automatiquement. La figure 2.1 montre les différences dans la classification des polarités de sentiments entre les deux approches : l'apprentissage automatique traditionnel (machine à vecteur de support (SVM), réseaux bayésiens ou arbres de décision) et l'apprentissage profond. Les réseaux de neurones artificiels et l'apprentissage profond fournissent actuellement les meilleures solutions à de nombreux problèmes dans les domaines de la reconnaissance d'images et de la parole, ainsi que dans le traitement du langage naturel. Plusieurs types de techniques d'apprentissage profond sont abordés dans cette section. [4]

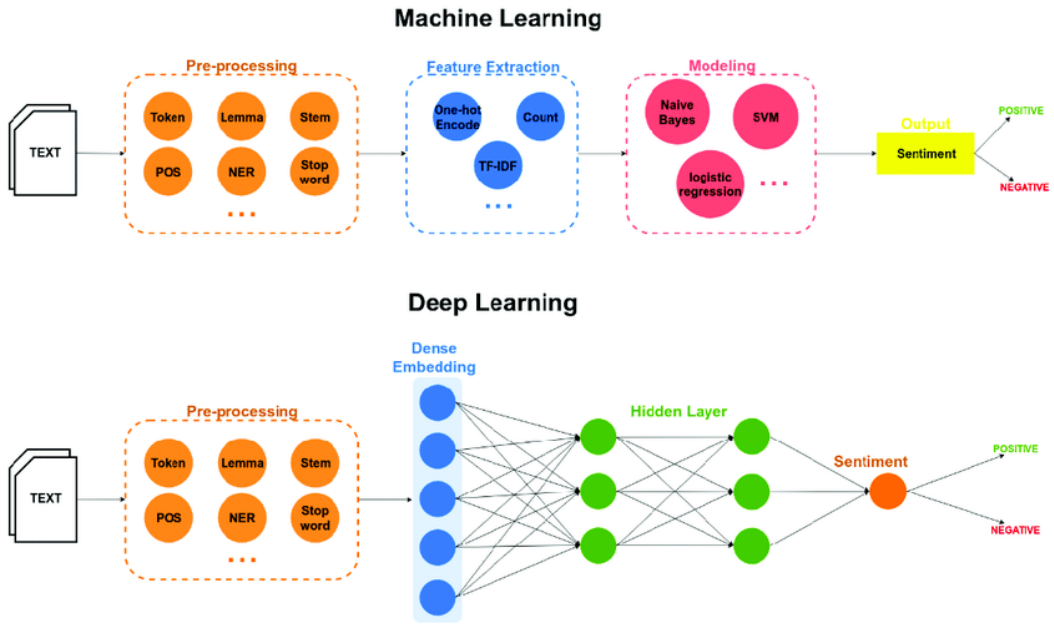

FIG. 2.1 : Différences entre deux approches de classification de la polarité des sentiments, l'apprentissage automatique (en haut) et l'apprentissage en profondeur (en bas). Partie du discours (POS) ; Reconnaissance d'entité nommée (NER) ;Terme Fréquence-Inverse Document Frequency (TF-IDF).[4]

2.3 réseaux de neurons :

2.3.1 Concepts de base

Les réseaux de neurones artificiels sont construits à partir d'un ensemble d'éléments de base appelés "neurones formels" qui sont un type d'unité. Ces neurones créent un graphe orienté lorsqu'ils sont appariés les uns aux autres. Les synapses sont représentées par les connexions entre les nœuds du graphique, qui sont similaires aux réseaux biologiques du cerveau. Les poids modifiés lors de la phase d'apprentissage sont utilisés par un algorithme spécifique pour pondérer ces relations. Les poids dans ce type d'algorithme sont ajustés pour réduire l'écart entre la sortie du réseau (hypothèse) et la sortie attendue (référence).[4]

2.3.2 Neurone formel

L'idée mathématique des "neurones formels" a été développée à partir de l'activité des neurones du système nerveux. Par le biais de connexions d'entrée, ces neurones reçoivent des informations d'autres nœuds. Les N valeurs d'entrée sont d'abord additionnées de manière pondérée par chaque neurone j. Les poids attribués aux entrées d'un neurone sont conservés dans une matrice w, où w_{ij} désigne le poids de la connexion d'entrée a_i du neurone j. La valeur seuil b_j, qui reflète la sortie d'un neurone "biaisé", est ajoutée à ce total. Le potentiel post-synaptique biaisé p_j est représenté par cette valeur totale, qui est

calculée comme suit :

$$p_j = \sum_{i=1}^{N} w_{ij} a_i + b_j \tag{2.1}$$

Enfin, une fonction d'activation f transforme ce potentiel biaisé pour obtenir la valeur d'activation

$$a_i = f(p_j) \tag{2.2}$$

de neurones qui peuvent ensuite passer à d'autres neurones.

Parmi les fonctions d'activation communément utilisées, nous pouvons citer la fonction sigmoïde qui fournit une sortie comprise entre 0 et 1 :

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

L'avantage de cette fonction est que sa dérivée, une composante indispensable aux algorithmes d'apprentissage, peut être obtenue facilement :

$$\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} = \sigma(x)(1 - \sigma(x)) \tag{2.4}$$

Perceptron Multicouches

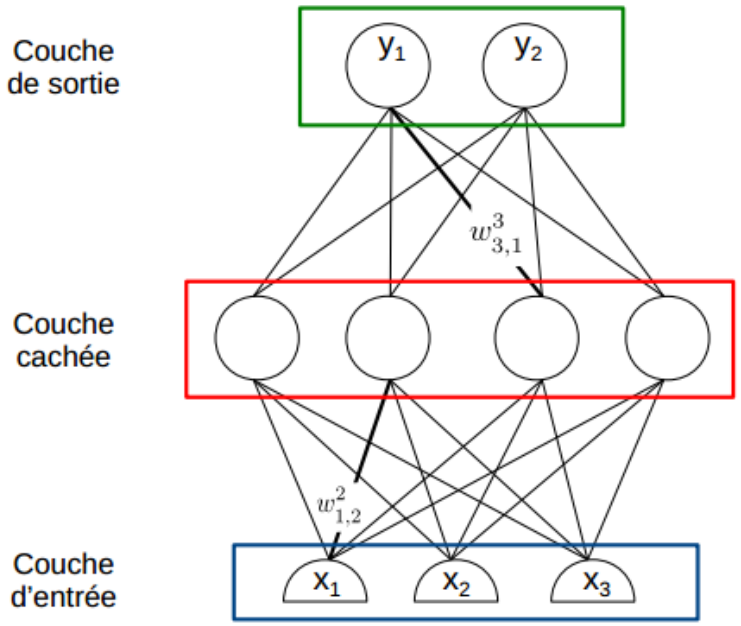

Fig. 2.2 : Un perceptron multicouches (MLP) à une seule couche cachée. Les neurones de la couche d'entrée sont représentés par des demi-cercles car ils ne déterminent pas des potentiels postsynaptiques biaisés.

Représentant des réseaux de neurones Feed-Forward Une classe de réseaux de neurones artificiels dont les nœuds forment un graphe Acyclique dirigé, où l'information circule dans

une seule direction, c'est-à-dire entrée pour sortir. Perceptron multicouche (Perceptron multicouche ou MLP), en La figure 2.2 est un réseau neuronal acyclique dont les nœuds sont organisés en trois niveaux ou plus, appelés "couches". Les couches adjacentes sont entièrement connectées, c'est-à-dire que les nœuds de chaque couche sont liés à tous les nœuds de la couche inférieure et à tous les nœuds de la couche supérieure. En revanche, il n'y a pas de connexions entre les cellules d'une même couche.
MLP se compose de trois types de couches qui sont une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie :

- La couche d'entrée est la première couche du réseau. l'activation de ce La couche reçoit les informations fournies par chaque vecteur d'entrée exemple. Par conséquent, la couche n'a pas de connexions d'entrée à partir de autres nœuds. D'autre part, il est entièrement connecté à la première couche caché.
- Dans un MLP à $N(N \geq 1)$ couches cachées, chaque couche $N-1$ Le caché inférieur est entièrement connecté au caché au-dessus. Cette La $N-$ième et dernière couche cachée est entièrement connectée à la couche de sortie.
- Les activations des neurones de la couche de sortie représentent les valeurs vectorielles de sortie du MLP

Les MLP sont couramment utilisés pour les tâches de classification [3] [29], mais moins fréquemment lorsqu'il s'agit de données séquentielles [39]. Par conséquent, dans le manuscrit ci-dessous, nous considérons que les MLP entrent dans la catégorie des algorithmes classiques, bien que ces modèles aient d'autres caractéristiques. Étant donné que ces réseaux sont paramétrés en ajustant les poids d'entrée des neurones, les MLP sont capables d'imiter le comportement de diverses fonctions. [19] ont même démontré qu'un MLP contenant une seule couche cachée et suffisamment de neurones avec des activations non linéaires peut approximer n'importe quelle fonction continue. Par conséquent, MLP est considéré comme une fonction générale. De plus, ces réseaux peuvent être utilisés pour générer des représentations de données plus robustes.

Rétropropagation du gradient

La phase d'apprentissage du MLP consiste à ajuster les poids des connexions en fonction des erreurs de prédiction constatées à chaque catégorisation de cas ultérieure. L'approche la plus utilisée pour ajuster les poids est la rétropropagation [11]. En partant de la dernière couche et en poursuivant jusqu'à la dernière couche pour arriver à la première couche cachée, cette technique trouve le gradient de l'erreur pour chaque neurone du réseau.

Le but de la mise à l'échelle du gradient inverse est d'ajuster les poids des liens pour réduire l'erreur quadratique

$$E = \frac{1}{2} \sum_{i=1}^{N} (ref_i - hyp_i)^2 \tag{2.5}$$

qui désigne la différence entre la sortie prédite (référence) et la sortie du réseau (hypothèse) pour un vecteur d'entrée particulier. La taille des vecteurs de sortie est représentée par le nombre N.

La rétropropagation par gradient est un problème d'optimisation pour lequel la méthode de descente de gradient peut être utilisée. Comme l'erreur E est une fonction des poids w, un minimum local peut être trouvé en déplaçant les poids dans la direction opposée du gradient $\frac{\partial E}{\partial w}$ multiplié par le taux d'apprentissage α :

$$\Delta w_{ij}^{*} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{2.6}$$

$$w_{ij}^{*} = w_{ij} + \Delta w_{ij}^{*} \tag{2.7}$$

Prenons un réseau de neurones MLP contenant L couches. La couche $k(2 \leq k \leq L)$ contient N_k neurones à laquelle est affecté un vecteur de biais b^k et une matrice de poids w^k. Dans cette matrice, un élément w_{ij}^k représente le poids de la connexion partant du neurone $i\,(1 \leq i \leq N_{k-1})$ de la couche $k-1$ vers le neurone $j\,(1 \leq j \leq N_k)$ de la couche k. Pour chaque nouvelle instance, une passe d'apprentissage est effectuée en 3 étapes :

1. **Propagation vers l'avant** : Le vecteur d'entrée est copié dans les activations a_i^1 de la première couche. Ensuite, pour chacune des k couches, partant de la première couche cachée et montant jusqu'à la couche de sortie, le potentiel biaisé p_i^k et l'activation a_i^k du neurone i sont calculés.

2. **Propagation vers l'arrière** : Calculer la dérivée Δ_i^L de l'erreur E pour l'activation a_i^L du neurone i de la couche de sortie :

$$\Delta_i^L = \left(ref_i - a_i^L\right) \frac{\partial f\left(p_i^L\right)}{\partial p_i^L} \tag{2.8}$$

   Ensuite, en descendant à partir de la dernière couche cachée $h = L-1$ jusqu'à la première $h = 2$, calculer le terme Δ_i^h pour chaque neurone :

$$\Delta_i^h = \sum_{j=1}^{u_{h+1}} \Delta_j^{h+1} w_{ij}^{h+1} \frac{\partial f\left(p_i^h\right)}{\partial p_i^h} \tag{2.9}$$

3. **Adaptation des paramètres**. Mettre à jour le biais et les poids des nœuds j pour chaque couche k :

$$b_j^{k*} = b_j^k + \alpha \Delta_i^k \tag{2.10}$$

$$w_{ij}^k = w_{ij}^k + \alpha \Delta_i^k a_j^k \tag{2.11}$$

La phase d'apprentissage est nettement plus coûteuse que la phase de classification, qui consiste à simplement propager les données de chaque nouvelle instance vers l'avant, comme dans l'étape 1 du processus d'apprentissage, pour produire le résultat de la classification automatisée.

2.4 Réseaux de neurones récurrents (RNN)

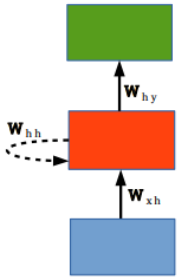

FIG. 2.3 : Représentation compacte des RNN. Toutes les flèches représentent des connexions complètes. La flèche en pointillée représente les connexions ayant un décalage temporel $(t-1)$.

Les réseaux neuronaux récurrents (Recurrent neural networks ou RNN), comme le montre la figure 2.3, contiennent des cycles à l'intérieur du graphe neuronal, contrairement aux MLP [7]. La capacité à modifier des séquences de vecteurs d'entrée, chacune représentant un événement, plutôt que des données isolées sans signification temporelle, est la principale motivation de ce type de conception. Ce type de réseau peut être considéré comme une série temporelle de réseaux MLP reliés par une séquence temporelle de réseaux MLP reliés entre eux par leurs couches cachées respectives en déroulant la modélisation compacte d'un RNN par rapport au temps (voir figure 2.4). Les RNN peuvent encoder des relations latentes entre les événements d'une séquence de vecteurs d'entrée grâce à ce lien.

Suivant cette modélisation, un RNN prend en entrée une séquence d'événements $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ et définit la séquence d'états cachés $\mathbf{h} = (h_1, h_2, \ldots, h_T)$ pour produire la séquence de vecteurs de sortie $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ en itérant de $t = 1$ à T :

$$h_t = \mathcal{H}\left(\mathbf{W}_{xh}x_t + \mathbf{W}_{hh}h_{t-1} + b_h\right) \tag{2.12}$$

$$y_t = \mathbf{W}_{hy}h_t + b_y \tag{2.13}$$

où T est le nombre total de vecteurs d'entrée, $\mathbf{W}_{\alpha\beta}$ est la matrice de poids entre les couches α et β, et b_β est le vecteur de biais de la couche β. La fonction $\mathcal{H}$ utilisée dans le cas des RNN est généralement la tangente hyperbolique (tanh).

La technique de rétropropagation par gradient, qui était destinée aux réseaux non bouclés, est insuffisante pour rendre compte des liens temporels. Considérons une représentation "hiérarchique" dépliée des RNN comme une solution à ce problème. L'échelle de temps, indiquée par les arcs diagonaux, a maintenant une signification hiérarchique dans la schématisation fournie par la figure 2.4, dans le sens où les couches cibles sont plus importantes. un standard élevé Nous pouvons appliquer la technique de rétropropagation par gradient adaptée aux réseaux de neurones non bouclés en raison de sa représentation

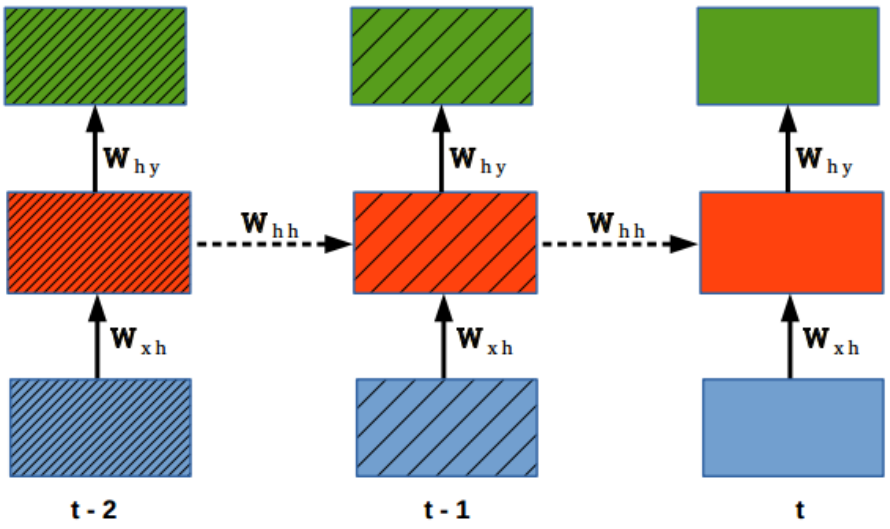

FIG. 2.4 : Représentation dépliée des RNN

hiérarchique. Les réseaux de neurones sont un type d'intelligence artificielle. La rétropropagation à travers le temps (Backpropagation Through Time ou BPTT) est le nom donné à cette variante [28].

L'inclusion de nombreux paramètres communs distingue cette représentation d'un réseau neuronal non bouclé "traditionnel". L'existence de nombreux paramètres communs dans un réseau neuronal. L'information est envoyée à travers les arcs diagonaux en utilisant une matrice de poids commune W_{hh}, par exemple. En outre, les arcs verticaux indiquent les matrices de poids W_{xh} et W_{hy}, qui sont partagées dans le temps.

L'algorithme d'apprentissage basé sur le BPTT contient trois phases basées sur ce qui précède :

- **Propagation vers l'avant :** De bas en haut, l'information se déplace comme elle le ferait dans un réseau non bouclé typique. La valeur de l'état caché précédent (h_{t-1}) ainsi que le vecteur d'entrée du temps $t(x_t)$ sont utilisés pour calculer le nouvel état caché h_t à chaque temps t (allant de1 à T) (voir formule 2.12). Le vecteur de sortie y_t est calculé à partir de ce dernier (voir formule 2.13).

- **Propagation vers l'arrière :** Pour chaque instant $t \in [1..T]$, le terme $\Delta_i^L(t)$du nœeud de sortie i est déterminé comme suit :

$$\Delta_i^L(t) = \left(\text{re}\, f_i(t) - a_i^L(t)\right) \frac{\partial f\left(a_i^L(t)\right)}{\partial a_i^L(t)} \tag{2.14}$$

  où les sorties prédites et produites du noeud i de la couche de sortie au temps t sont re $f_i(t)$ et $a_i^L(t)$, respectivement.
  Soit Δ_j le terme d'erreur correspondant au nœud j du niveau supérieur pour chaque temps t (allant de T à 1) (selon la hiérarchie de la figure 2.5), C'est-à-dire, correspondant à la couche de sortie du temps t ou à la couche cachée du temps $t+1$. Le poids du lien entre le neurone I de la couche cachée du temps t et le neurone j est

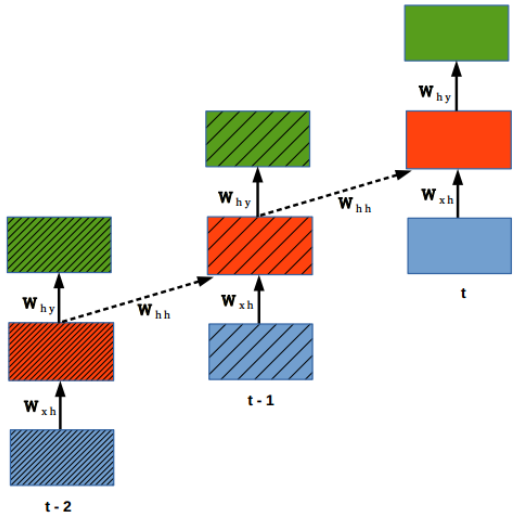

FIG. 2.5 : Représentation dépliée hiérarchisée des RNN

représenté par chaque membre w_{ij} de la matrice w^h ou de la matrice w^y. La formule suivante est utilisée pour calculer le terme $\delta_i^h(t)$ du neurone i.

$$\Delta_i^h(t) = \left(\sum_j \Delta_j w_{ij}\right) \frac{\partial f\left(a_i^h(t)\right)}{\partial a_i^h(t)} \tag{2.15}$$

- **Adaptation des paramètres.**Mettez à jour le biais et les poids de chaque neurone j dans une couche cachée ou couche l qui reçoit des connexions des neurones i dans la couche k. Le biais et les poids doivent être mis à jour comme suit :

$$b_j^{l*} = b_j^l + \alpha \sum_{\tau=0}^{t} \Delta_i^k(\tau) \tag{2.16}$$

$$w_{ij}^{l*} = w_{ij}^l + \alpha \sum_{\tau=0}^{t} \Delta_i^k(\tau) a_j^l(\tau) \tag{2.17}$$

où α est le taux d'apprentissage, $\delta_i^k(\tau)$ est l'erreur du nœud i de la couche k del'instant τ et al $j(\tau)$ est l'activation de l'unité j de la couche l de l'instant τ.

2.4.1 Long Short-Term Memory (LSTM)

Les RNN ont l'avantage de pouvoir analyser les données actuelles tout en tenant compte du contexte antérieur. Cependant, ces réseaux peinent à traiter des séquences un peu longues, notamment celles qui comptent plus de dix occurrences [32]. En effet, l'erreur produite par la rétropropagation du gradient diminue ou, plus rarement, croît de façon exponentielle par rapport à l'échelle de temps dans les calculs cumulatifs à long terme. Les termes "vanishing gradient" et "exploding gradient" sont utilisés pour décrire ces deux difficultés [32]. Comme nous l'avons découvert, ce problème se pose également dans les conceptions profondes non bouclées. Dans cette situation, le gradient de dissipation ou d'éclatement s'aggrave lorsque le nombre de couches augmente.

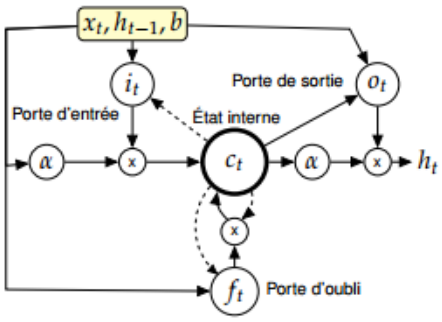

FIG. 2.6 : Une cellule Long Short-Term Memory (LSTM). Les flèches en pointillé représentent les opérandes avec un décalage temporel ($t-1$). La fonction d'activation α (pour l'entrée et la sortie) est généralement la tangente hyperbolique tanh.

Une extension de l'idée des RNN, en particulier l'architecture de la mémoire à long terme (LSTM), est l'une des solutions les plus efficaces à ce problème de calcul du gradient (Hochreiter et Schmidhuber, 1997). Les LSTM sont un type de RNN dont la conception générale et la formulation mathématique sont identiques à celles de la figure 2.3 et des formules 2.12 et 2.13.

La façon dont l'état caché est conservé est ce qui rend les LSTM uniques. Le traitement récurrent, désigné par la fonction $\mathcal{H}$, est donné par une simple fonction *tanh* dans le cas des RNN de base. Ce traitement est remplacé dans les LSTM par une "cellule mémoire", comme le montre la figure 2.6, qui remplace la couche cachée de la figure 2.3. La cellule LSTM possède un nœud central qui contient l'état interne de la cellule (ou mémoire), ainsi qu'un certain nombre de "portes" classées en trois types. Ces portes permettent de contrôler le stockage des informations séquentielles en mémoire (portes d'entrée et d'oubli), ainsi que la fonction de l'état interne dans la génération de chaque sortie (portes de sortie).Les nouveaux événements sont moins pris en compte dans les informations de la cellule lorsque la porte d'entrée est fermée, par exemple.

La figure 2.6 illustre l'une des variantes de LSTM les plus utilisées, qui est complétée par des connexions de type peephole [13]. Les peepholes sont des connexions supplémentaires qui permettent aux différentes portes d'être informées du contenu de l'état interne avant de déterminer si elles doivent s'ouvrir ou se fermer. Les portes d'entrée et d'oubli sont calculées avant que l'état interne ne soit modifié, comme spécifié. Ces portes consultent l'état interne précédent pendant la "propagation vers l'avant" à un instant t. (celui de l'instant $t-1$)

Selon cette représentation, $\mathcal{H}$ est désormais une fonction composite définie par :

$$i_t = \sigma\left(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{W}_{ci}c_{t-1} + b_i\right) \tag{2.18}$$

$$f_t = \sigma\left(\mathbf{W}_{xf}x_t + \mathbf{W}_{hf}h_{t-1} + \mathbf{W}_{cf}c_{t-1} + b_f\right) \tag{2.19}$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(\mathbf{W}_{xc}x_t + \mathbf{W}_{hc}h_{t-1} + b_c\right) \tag{2.20}$$

$$o_t = \sigma\left(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{W}_{co}c_t + b_o\right) \tag{2.21}$$

$$h_t = o_t \tanh\left(c_t\right) \tag{2.22}$$

où i, f et o représentent les portes d'entrée, d'oubli et de sortie, respectivement, et c représente le vecteur d'état interne de même taille que le vecteur caché h. Les matrices $\mathbf{W}$ de la cellule c aux portes i, f, et o sont diagonales, ainsi un élément j dans le vecteur de chaque porte ne reçoit que l'élément j du vecteur de la cellule. Enfin, la fonction logistique sigmoïde est désignée par σ.

Les LSTM ont démontré leur efficacité dans un large éventail d'applications. Dans de nombreuses tâches traitant de données séquentielles, ils sont désormais considérés comme la solution de pointe [15] [20] [18] [21]. Leur contribution est plus notable dans le cas de séquences d'événements prolongées [38] .

2.4.2 Long Short-Term Memory Bidirectionnels (BLSTM)

Avant de traiter l'élément suivant dans une séquence, les réseaux récurrents n'utilisent que le contexte précédent. Cependant, la compréhension du contexte futur par rapport à un certain moment peut être très utile dans certains types d'applications. C'est le cas, par exemple, dans le traitement des séquences de mots.

D'une part, l'existence du mot it aide à la prédiction du mot suivant, à savoir is, par opposition à d'autres formes conjuguées comme are ou aren't. Les RNN "unidirectionnels" assurent ce flux d'informations dit "vers l'avant". L'existence du mot arrive, d'autre part, peut également transmettre des informations concernant le deuxième mot. Par exemple, le verbe être est plus approprié que le mot avoir. Les RNN unidirectionnels, en revanche, ne le peuvent pas. n'assurent pas le transfert d'informations dans la direction dite "inverse".

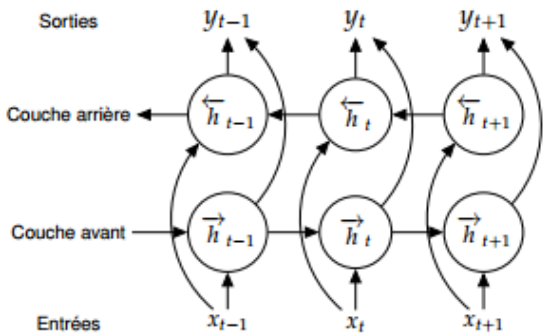

FIG. 2.7 : Un RNN bidirectionnel (Bidirectional RNN ou BRNN).

Cette exigence peut être observée dans le marquage morphosyntaxique, la traduction automatique et la reconnaissance de l'écriture manuscrite, entre autres applications TALN.

La figure 2.7 présente des réseaux neuronaux récurrents bidirectionnels (BRNN), qui peuvent traiter des données séquentielles dans les deux sens en utilisant deux couches cachées différentes [26]. (une pour chaque direction). employer deux couches cachées indépendantes (une pour chaque direction) pour séquencer les données dans les deux directions En itérant $-h$ à partir des couches cachées " avant " (-h) et " arrière " (- h), cette forme de RNN offre les données des couches cachées " avant " (-h) et " arrière " (- h) à une seule couche de sortie y.

Les réseaux de neurones récurrents bidirectionnels (Bidirectional kNN ou BRNN) [24], présentés dans la figure 2.7 peuvent traiter les données séquentielles dans les deux sens en utilisant deux couches cachées séparées (une pour chaque sens). Ce type de RNN fournit, à une seule couche de sortie $\mathbf{y}$, des données provenant des deux couches cachées « avant » ($\overrightarrow{\mathbf{h}}$) et « arrière » ($\overleftarrow{\mathbf{h}}$) en itérant $\overrightarrow{\mathbf{h}}$ de $t = 1$ à T et $\overleftarrow{\mathbf{h}}$ de $t = T$ à 1 :

$$\overleftarrow{h}_t = \mathcal{H}\left(\mathbf{W}_x \overleftarrow{h} x_t + \mathbf{W}_{\overleftarrow{h}\,\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right) \tag{2.23}$$

$$\overrightarrow{h}_t = \mathcal{H}\left(\mathbf{W}_x \overrightarrow{h} x_t + \mathbf{W}_{\overrightarrow{h}\,\overrightarrow{h}} \overrightarrow{h}_{t+1} + b_{\overrightarrow{h}}\right) \tag{2.24}$$

$$y_t = \mathbf{W}_{\vec{h}y}\vec{h}_t + \mathbf{W}_{\overleftarrow{h}\,y}\overleftarrow{h}_t + b_y \tag{2.25}$$

L'algorithme BPTT est utilisé pour l'apprentissage des BRNN. La seule distinction est que le flux d'information est indépendant dans les deux directions (de 1 à T et de T à 1). Les paramètres du réseau sont appris à l'aide d'un algorithme [5] [31] :

1. **Propagation vers l'avant :**Prévoir la sortie en analysant les données d'entrée dans les deux directions du BRNN :
   - Pour t de 1 à T :
     - Faites une passe avant pour la couche cachée au premier plan $\overleftarrow{h}$.
   - Pour T de T à 1 :
     - Faites la passe avant de la couche arrièr $\overrightarrow{h}$.
   - Pour tout $t \in [1..T]$:
     - En utilisant les résultats de l'activation des deux couches cachées, faites la passe avant pour la couche de sortie.
2. **Propagation vers l'arrière :**Déterminer les termes d'erreur Δ :
   - Pour tout $t \in [1..T]$:
     - Créez la passe arrière de la couche de sortie.
   - Pour T de T à 1 :
     - Un retour vers la couche cachée est effectué avant $\overrightarrow{h}$ en utilisant le terme Δ de la couche de sortie.
   - Pour t de 1 à T :
     - Effectuez une passe arrière vers la couche cachée arrière $\overleftarrow{h}$ en utilisant le terme Δ de la couche de sortie.

3. **Adaptation des paramètres.**

Avec la capacité de modéliser des contextes passés et futurs, les BRNN ont depuis émergé dans diverses applications telles que le traitement de la parole [14], la classification des protéines (Chen et Chaudhari, 2004) et la synthèse vocale (Fan et al.) déjà fait ses preuves. et al., 2014).

En remplaçant les deux couches récurrentes du BRNN par des cellules LSTM (voir Figure 2.6), un LSTM bidirectionnel (Bidirectional LSTM ou BLSTM) [31]. BLSTM peut transporter un long contexte et tirer parti de Les deux sens de la structure. Par conséquent, le vecteur de sortie y est obtenu en traitant Séquences d'entrée simultanées, via la fonction composite H Équation 3.51, dans les deux sens ($\overrightarrow{h}$ et $\overleftarrow{h}$).

2.5 Conclusion

À la fin de ce chapitre, nous avons discuté des réseaux de neurones récurrents Types de LSTM et leur compréhension particulière du traitement séquentiel des données. Cette méthode diffère des autres méthodes adaptées aux séquences avec des fonctions différentes. En fait, en raison de sa structure en couches de neurones, les LSTM peuvent être utilisés pour générer des représentations vectorielles à partir de fichiers de données brutes sérialisés. Ces nouvelles représentations sont agencées sous forme de vecteurs de caractéristiques, qui peuvent être mieux assimilés, facilitant le traitement du langage naturel.

Chapitre 3

Le modèle de transformateur pour l'analyse des sentiments

3.1 Introduction

Les modèles de représentation textuelle contextualisée pré-entraînés ont permis des avancées massives dans les tâches de compréhension du langage naturel (NLP), et ont atteint des performances dans de multiples tâches NLP. [17] ;[9]. Les premiers modèles de représentation de texte pré-entraînés visaient à représenter les mots en captant leurs propriétés syntaxiques et sémantiques distribuées à l'aide de techniques telles que Word2vec [23] et GloVe [25]. Cependant, ces modèles n'ont pas incorporé le contexte dans lequel un mot apparaît dans son incorporation.

Récemment, l'accent a été mis sur l'application de l'apprentissage par transfert en affinant des modèles langage pré-entraînés de grande taille pour des tâches NLP en aval avec un nombre relativement faible d'exemples. pour des tâches NLP avec un nombre relativement faible d'exemples, ce qui entraîne une amélioration notable des performances pour ces tâches. Cette approche tire parti de modèles de langage qui ont été pré-entraînés de manière non supervisée (ou parfois appelée auto-supervisée). Cependant, cet avantage présente des inconvénients, notamment les énormes corpus nécessaires au pré-entraînement, en plus du coût élevé des jours de calcul nécessaires à l'entraînement (les derniers modèles ont nécessité plus de 500 TP). formation (les derniers modèles ont nécessité plus de 500 TPU ou GPU pendant des semaines [6] ; [27] ;[1]. Ces inconvénients ont limité la disponibilité de tels modèles à l'anglais principalement et à une poignée d'autres langues. Pour remédier à cette lacune, des modèles multilingues ont été formés pour apprendre des représentations pour plus de 100 langues simultanément, mais ils sont toujours en retard sur les modèles monolingues en raison de la faible représentation des données et de la petite taille des données.

Pour une vocabulaire spécifique à la langue. Si les langues dont la structure et le vocabulaire sont similaires peuvent bénéficier des représentations partagées [6], ce n'est pas le cas pour d'autres langues, comme l'arabe, qui diffèrent par leur structure morphologique et syntaxique et partagent très peu avec d'autres langues abondantes basées sur le latin. langues à base latine.

Nous décrivons le processus de pré-entraînement du modèle de transformateur BERT [9] pour le langage Arabic, que nous appellons ARABERT. Qu'est-ce qu'ARABERT, le mécanisme utilisé et les applications qui lui ont été appliquées ?

3.2 Le Transformateur

Transformer en anglais est basé uniquement sur le mécanisme de l'attention, et se passe entièrement de réseaux récurrents et convolutifs. À l'heure actuelle, ce modèle a fait l'objet d'une attention particulière. et joue un rôle clé dans de nombreux modèles neuronaux de langage, tels que BERT, GPT et Universal Transformer.[22]

Architecture modèle

La plupart des modèles compétitifs de traduction de séquences neuronales ont une structure encodeur-décodeur . Ici, le codeur fait correspondre une séquence d'entrée de représentations de symboles $(x1, ..., xn)$ à une séquence de représentations continues $z = (z1, ..., zn)$. Étant donné z, le décodeur génère alors une séquence de sortie séquence de sortie $(y1, ..., ym)$ de symboles, un élément à la fois. À chaque étape, le modèle est auto-régressif, consommant les symboles générés précédemment comme entrée supplémentaire lors de la génération du suivant.
Transformer suit cette architecture globale en utilisant des couches empilées d'auto-attention et des couches ponctuelles entièrement connectées pour l'encodeur et le décodeur, présentées respectivement dans les moitiés gauche et droite .[35]

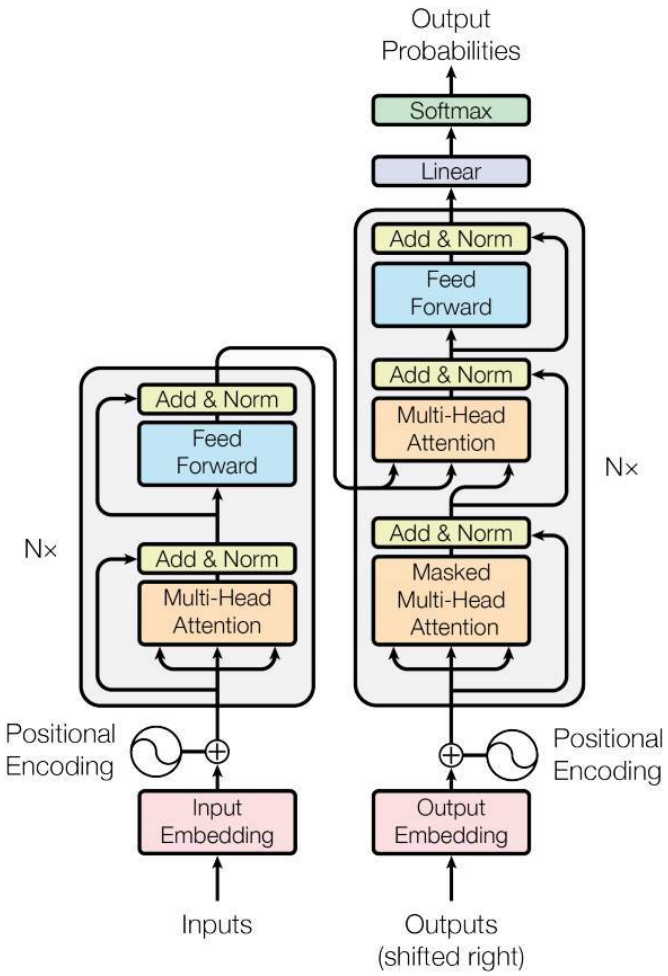

Fig. 3.1 : Le Transformateur - modèle d'architecture [35]

3.3 L'Attention

Une fonction d'attention peut être décrite comme la mise en correspondance d'une requête et d'un ensemble de paires clé-valeur avec une sortie, où la requête, les clés, les

valeurs et la sortie sont toutes des vecteurs. La sortie est calculée comme une somme pondérée des valeurs, où le poids attribué à chaque valeur est calculé par une fonction de compatibilité de la requête avec la clé correspondante. [35]

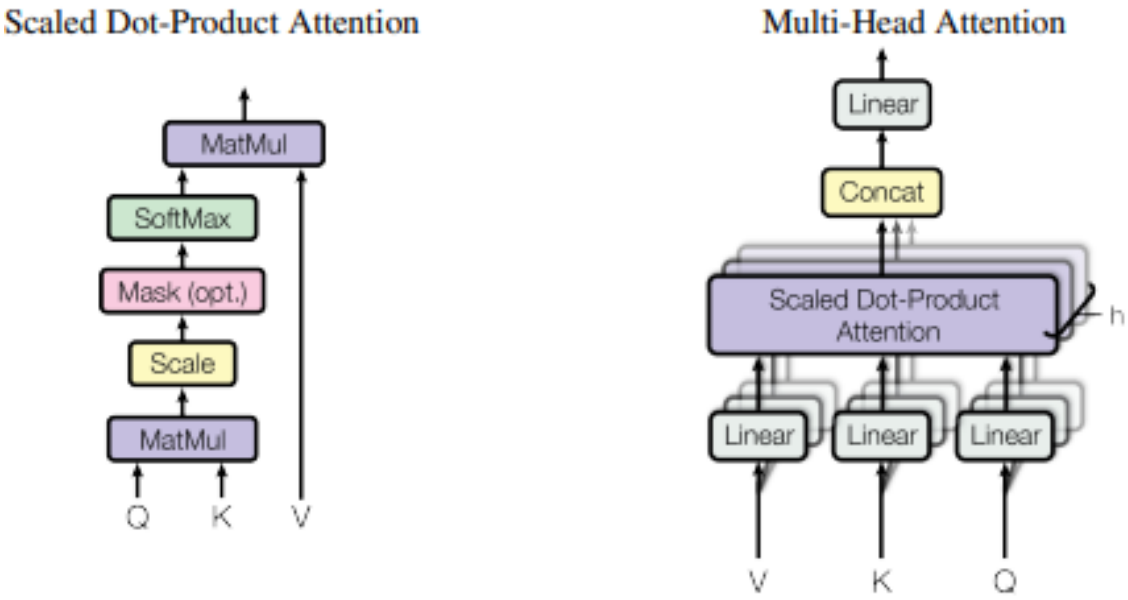

Fig. 3.2 : Le Transformateur - modèle d'architecture [35]

3.4 BERT

Nous présentons BERT(english :Bidirectional Encoder Representations Transformers) et son implémentation détaillée dans cette section. Le modele comprend deux étapes : la pré-formation et le réglage fin. Pendant le pré-entraînement, le modèle est entraîné sur des données non étiquetées au cours de différentes tâches de pré-entraînement. Pour l'ajustement fin, le modèle BERT est d'abord initialisé avec les paramètres pré-entraînés, et tous les paramètres sont ajustés en utilisant des données étiquetées à partir des tâches en aval. Chaque tâche en aval dispose de modèles affinés distincts, même s'ils sont initialisés avec les mêmes paramètres pré-entraînés. L'exemple de réponse à une question de la figure 3.3 [9].

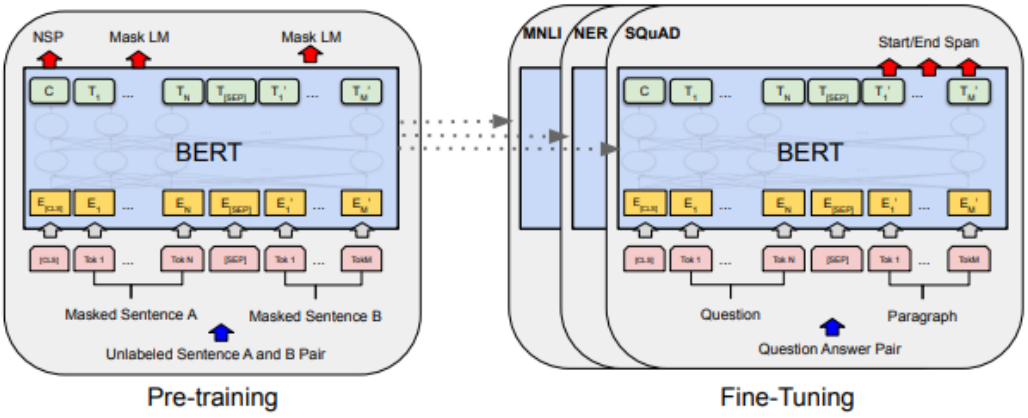

FIG. 3.3 : Procédures globales de pré-entraînement et d'ajustement fin pour BERT. A part les couches de sortie, les mêmes architectures sont utilisées pour le pré-entraînement et le réglage fin. Les mêmes paramètres de modèle pré-entraînés sont utilisés pour initialiser les modèles pour différentes tâches en aval. modèles pour différentes tâches en aval. Lors de l'ajustement fin, tous les paramètres sont ajustés. [CLS] est un symbole spécial spécial ajouté devant chaque exemple d'entrée, et [SEP] est un symbole spécial de séparation (par exemple, pour séparer les questions des réponses).[9]

AraBERT

AraBERT, est un modèle de représentation de la langue arabe pour améliorer l'état de l'art dans plusieurs tâches NLP arabes. La creation d' ARABERT sur la base du modèle modèle BERT, un encodeur transformateur bidirectionnel empilé. Ce modèle est largement considéré comme la base de la plupart des résultats de l'état de l'art dans différentes tâches NLP dans plusieurs langues.La configuration BERTbase, qui comporte 12 blocs d'encodeurs, 768 dimensions cachées, 12 têtes d'attention, 512 dimensions maximales, etc. cachées, 12 têtes d'attention, une longueur maximale de séquence de 512 et un total de 110M paramètres. Nous avons également introduit un prétraitement supplémentaire avant le pré-entraînement du modèle, afin de mieux s'adapter à la langue arabe.[2]

Ensemble de données

Le BERT original a été entraîné sur 3,3 milliards de mots extraits de la Wikipédia anglaise et du Corpus des livres . Puisque les vidages de Wikipedia en arabe sont petits comparés à ceux de l'anglais, nous avons manuellement manuellement des sites d'information arabes pour trouver des articles. En outre, nous avons En outre, nous avons utilisé deux grands corpus arabes accessibles au public : (1) le Corpus arabe de 1,5 milliard de mots , qui est un un corpus contemporain comprenant plus de 5 millions d'articles extraits de d'articles extraits de dix sources d'information majeures couvrant 8 pays. pays, et (2) OSIAN : the Open Source International Arabic News Corpus . Arabic News Corpus qui se compose de 3,5 millions d'articles (1B tokens) provenant de 31 sources d'actualités dans 24 pays arabes. La taille finale de l'ensemble de données de pré-entraînement, après

la suppression des phrases dupliquées, est de 70 millions de phrases, ce qui correspond à à 24GB de texte. Ce jeu de données couvre les actualités de différents médias dans différentes régions arabes, et peut donc être donc être représentatif d'un large éventail de sujets discutés dans le monde arabe. Il est intéressant de mentionner que nous avons préservé les mots qui comprennent des caractères latins, puisqu'il est courant de mentionner des les entités nommées, les termes scientifiques ou techniques dans leur langue langue d'origine, afin d'éviter toute perte d'information.[2]

Les applications de modele

- **Question reponse (Question Answering QA) :**Dans le QA, étant donné une question et un passage contenant la réponse, le modèle doit sélectionner un intervalle de texte qui contient les réponses. Ceci est fait en prédisant un jeton de "début" et un jeton de "fin" à condition que le jeton de "fin" apparaisse après le jeton de "début". que le token "fin" apparaisse après le token "début". Pendant l'entraînement, l'intégration finale de chaque token dans le passage est introduit dans deux classificateurs, chacun avec un seul ensemble de poids, qui sont appliqués à chaque mot. de poids, qui sont appliqués à chaque token. Le produit scalaire des incorporations de sortie et du classificateur est ensuite introduit dans une couche softmax pour produire une distribution de probabilité sur tous les jetons. Le jeton ayant la plus grande probabilité d'être un jeton de "début" est ensuite sélectionné, et le même processus est répété pour le jeton de "fin".

- **Reconnaissance d'entité nommée (Named Entity Recognition) :**Pour la tâche NER, chaque token de la phrase est étiqueté avec le format IOB2, où l'étiquette "B" correspond au premier mot de l'entité, l'étiquette "I" correspond au reste des mots de la même entité, et l'étiquette "O" indique que le mot étiqueté n'est pas une entité nommée souhaitée. Par conséquent, nous traitons le système comme un processus de classification multi-classes, ce qui nous permet d'utiliser certaines méthodes de classification de texte pour étiqueter les tokens. En outre, après avoir utilisé le tokenizer AraBERT, nous n'entrons dans le modèle que le premier sous-titre de chaque mot. de chaque mot au modèle.

- **Sequence Classification :** Pour affiner AraBERT pour la classification des séquences, nous prenons l'état caché final du premier token, qui correspond à l'intégration du mot du token spécial "[CLS]" ajouté au début de chaque phrase. phrase. Nous ajoutons ensuite une simple couche de feed-forward avec Softmax standard pour obtenir la distribution de probabilité sur les classes de sortie prédites. les classes de sortie prédites. Pendant le réglage fin, le classificateur et les poids du modèle pré-entraîné sont entraînés conjointement pour maximiser la log-probabilité de la classe correcte.[2]

Après avoir abordé Transformateur, BERT et AraBERT, la conception de notre système a été présentée et les étapes importantes ont été suivies étape par étape. Maintenant dans cette section, nous presentons la préparation du Framework, que nous développons et ces étapes , et dans la dernière nous terminon par les résultats obtenus.

3.5 Les étapes de notre travail

Nous avons choisi l'analyse des sentiments, pour classer et évaluer les sentiments des utilisateurs en utilisant modèle transformateur. AraBERT propose un modèle basé sous le Réseau twitter. Le modèle est retrainer à la base d'un ensemble des données.

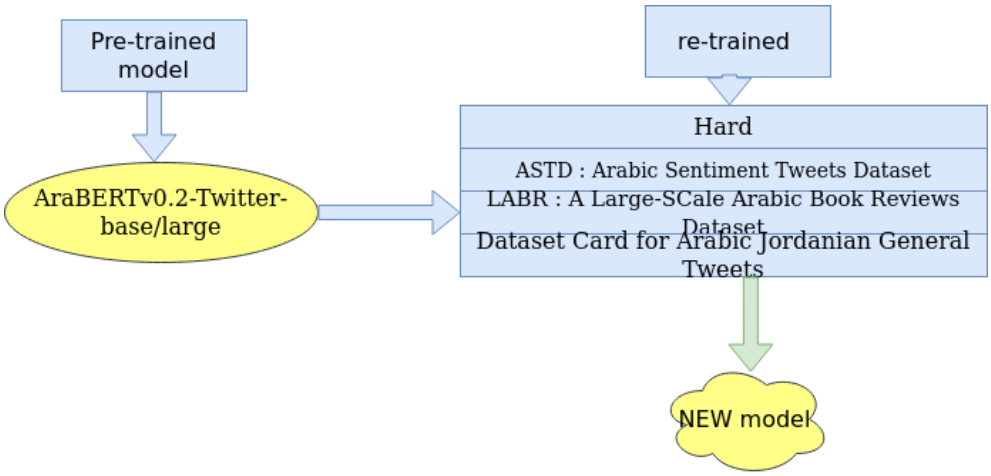

FIG. 3.4 : Les étapes de notre modèle

architecture AraBERT dans 'analyse des sentiments

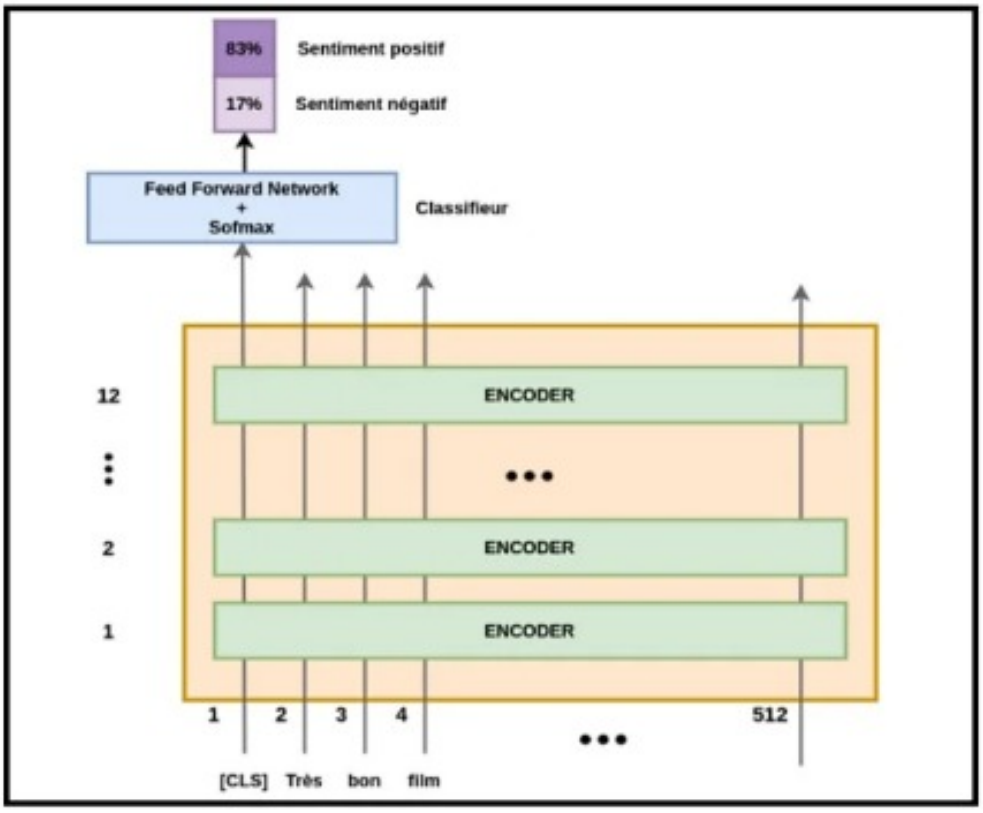

FIG. 3.5 : architecture du réglage fin

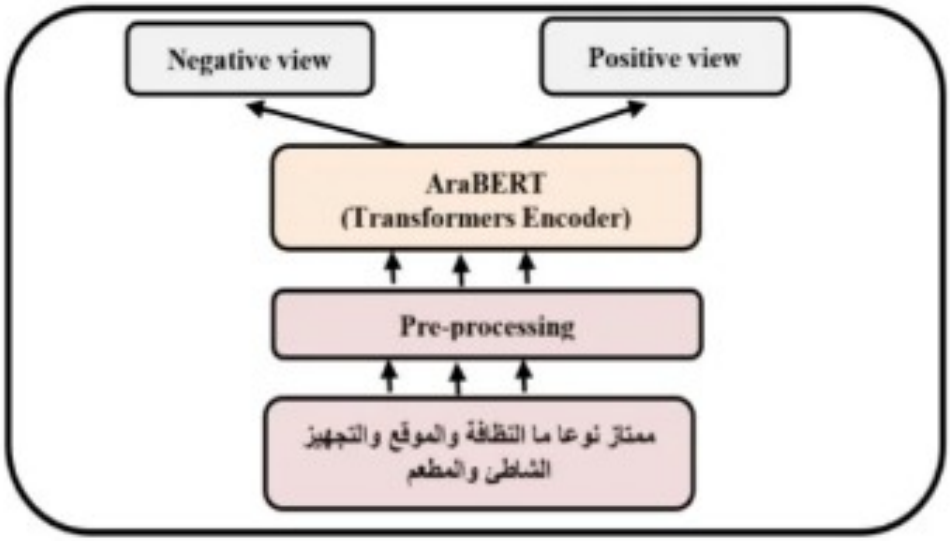

FIG. 3.6 : architecture AraBERT dans 'analyse des sentiments

AraBERTv0.2-Twitter-base/large

AraBERTv0.2-Twitter-base/large sont deux nouveaux modèles pour les dialectes et les tweets arabes, formés en pour suivant la pré-formation à l'aide de la tâche MLM sur 60M de tweets arabes (filtrés à partir d'une collection sur 100M).

FIG. 3.7 : Logo AraBERT

Les deux nouveaux modèles ont ajouté des emojies à leur vocabulaire en plus de mots communs qui n'étaient pas présents au départ. La pré-formation a été effectuée avec une longueur de phrase maximale de 64 seulement pour 1 époque.

AraBERT est un modèle de langue arabe pré-entraîné basé sur l'architecture BERT de Google. AraBERT utilise la même configuration BERT-Base. Plus de détails sont disponibles dans le document AraBERT et dans le Meetup AraBERT.

ASTD : Arabic Sentiment Tweets Dataset

Cet dataset contient plus de 10k tweets de sentiment arabe classés en quatre classes subjectives positives, subjectives négatives, subjectives mixtes et objectives. Deux ensembles d'expériences d'analyse des sentiments de base sont pris en charge avec l'ensemble de données.

LABR : A Large-SCale Arabic Book Reviews Dataset

Cet dataset contient plus de 63 000 critiques de livres en arabe. Il s'agit du plus grand ensemble de données d'analyse des sentiments pour l'arabe à ce jour. Les critiques de livres ont été recueillies sur le site Web Goodreads au cours du mois ou de mars 2013. Chaque critique de livre est accompagnée de l'identifiant de la critique goodreads, de l'identifiant de l'utilisateur, de l'identifiant du livre, de la note (1 à 5) et du texte de la critique.

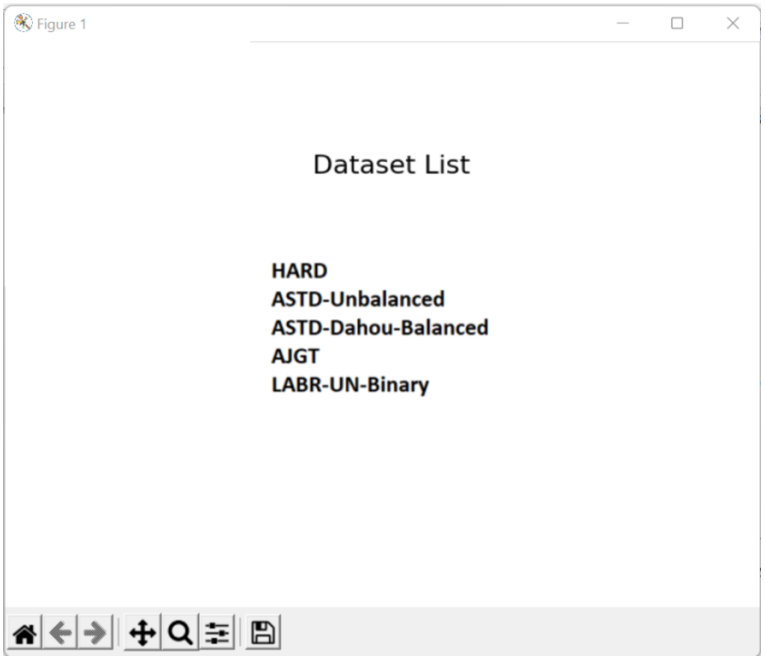

Fig. 3.8 : Répertorier l'ensemble de données

Dataset of Hard

Cet ensemble de données contient 93 700 avis d'hôtels en langue arabe. Les avis d'hôtels ont été recueillis sur le site Booking.com en juin/juillet 2016. Les avis sont exprimés en arabe standard moderne ainsi qu'en arabe dialectal. Le tableau suivant résume certaines statistiques sur l'ensemble de données HARD .

Dataset Card for Arabic Jordanian General Tweets

Le corpus AJGT (Arabe Jordanian General Tweets) consistait en 1 800 tweets annotés comme positifs et négatifs. Arabe standard moderne (MSA) ou dialecte jordanien.

L'image précédente est la liste des datasets que nous avons utilisées pour entraîner le modèle. Il contient un grand ensemble de données divisé en plusieurs dataset tel que HARD, ASTD-Unbalanced … etc.
Nous montrons comment réaliser les différentes étapes du système qui ont été expliquées précédement.
Nous allons utiliser le twitter AraBERT car il a des emojis et des dialects.

Fig. 3.9 : model AraBERT

3.6 Outils et Environnement de programmation

Afin de réaliser notre système, nous avons utilisé un ensemble d'outils et d'environnements de programmation, Nous citerons les plus importants d'entre eux :

3.6.1 Environnement matériel

L'application a été développée sur un PC Portable (HP modèle : HP EliteBook Computer) ayant les caractéristiques suivantes :

- Processeur Intel Core i5-7700 CPU@ 2.30GHz 2.29 GHz.
- Mémoire installée (RAM) : 16,00 Go.
- Disque Dure : 512 gb SSD.
- Carte graphique : Intel HD Graphics 630.
- Système d'exploitation : Windows 11 Professionnel 64 bit.

3.6.2 Environnement logiciel

Anaconda

Le Navigateur Anaconda est une interface graphique (GUI) incluse dans la distribution Anaconda, et qui permet aux utilisateurs de lancer des applications, mais aussi de gérer les librairies conda, les environnements et les canaux sans utiliser la moindre ligne de commande.

Le Navigateur peut également accéder à des librairies présentes sur le Cloud Anaconda ou dans un Repository Anaconda local, afin de les installer dans un environnement, les

exécuter et les mettre à jour. Il est disponible pour Windows, macOS et Linux.

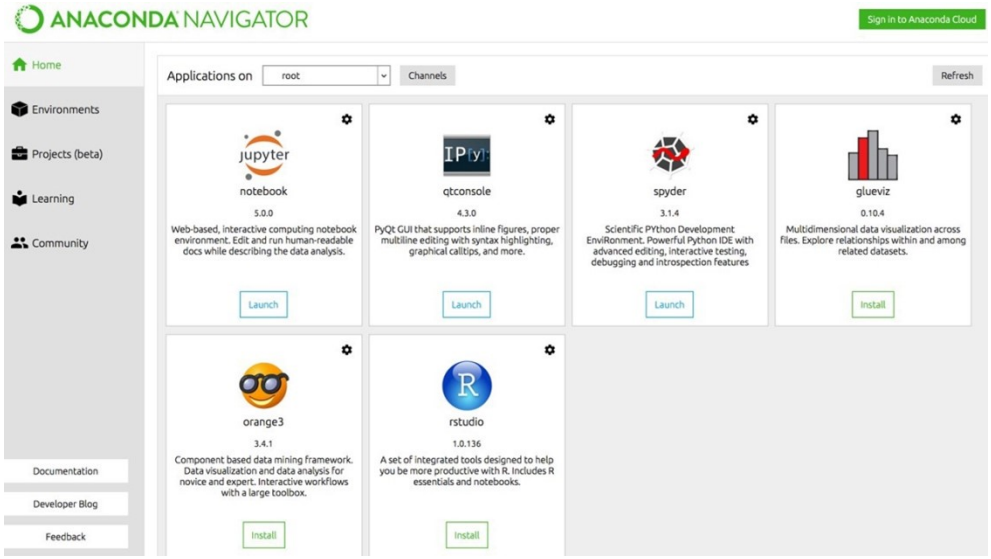

Fig. 3.10 : interface Anaconda

Les applications suivantes sont disponibles par défaut dans le navigateur :

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder

Pour valider notre application nous adopterons sur Python qui est un puissant langage de programmation polyvalent. Il est utilisé dans le développement Web, la science des données, la création de prototypes de logiciels, etc. Python a une syntaxe simple et facile à utiliser.
PyTorch est une bibliothèque logicielle Python open source d'apprentissage machine qui s'appuie sur Torch (en) développée par Meta8.
PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond (deep learning). Ces calculs sont optimisés et effectués soit par le processeur (CPU) soit, lorsque c'est possible, par un processeur graphique (GPU) supportant CUDA. Il est issu des équipes de recherche de Facebook, et avant cela de Ronan Collobert dans l'équipe de Samy Bengio9 à l'IDIAP.

PyTorch

PyTorch est dérivé d'un logiciel antérieur, Torch, qui s'utilisait avec le langage Lua. PyTorch est indépendant de Lua et se programme en Python.

FIG. 3.11 : Logo PyTorch

PyTorch permet de :
manipuler des tenseurs (tableaux multidimensionnels), de les échanger facilement avec Numpy et d'effectuer des calculs efficaces sur CPU ou GPU (par exemple, des produits de matrices ou des convolutions) ;calculer des gradients pour appliquer facilement des algorithmes d'optimisation par descente de gradient. PyTorch utilise la bibliothèque autograd.

Prétraitement

Avant de démarrer le processus de prétraitement des données, nous devons d'abord vérifier que la base de données est équilibrée afin que le nombre des instances pour chaque classe soit convergent. Le diagramme suivant montre que la base de données que nous avons utilisée est équilibrée de sorte que chaque classe.

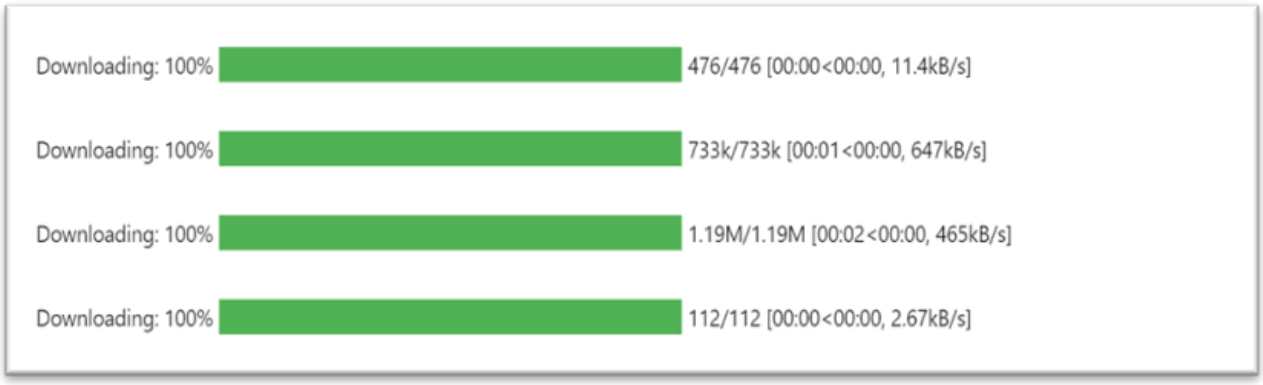

FIG. 3.12 : Télécharger modele

Longueurs des phrases d'entraînement : La longueur de séquence des indices de jeton est supérieure à la longueur de séquence maximale spécifiée pour ce modèle ($521 > 512$). L'exécution de cette séquence dans le modèle entraînera des erreurs d'indexation.
Les lignes de code responsables du processus de verification à la longueur de séquence maximale spécifiée pour ce modèle.

```
print("Training Sentence Lengths: ")
plt.hist([ len(tok.tokenize(sentence))
for sentence in selected_dataset.train[DATA_COLUMN].to_list()],bins=range(0,128,2))
plt.show()

print("Testing Sentence Lengths: ")
plt.hist([ len(tok.tokenize(sentence))
for sentence in selected_dataset.test[DATA_COLUMN].to_list()],bins=range(0,128,2))
plt.show()
✓ 7.1s
```

FIG. 3.13 : Vérifiez la longueur de la séquence du modèle

Longueurs des phrases d'entraînement :

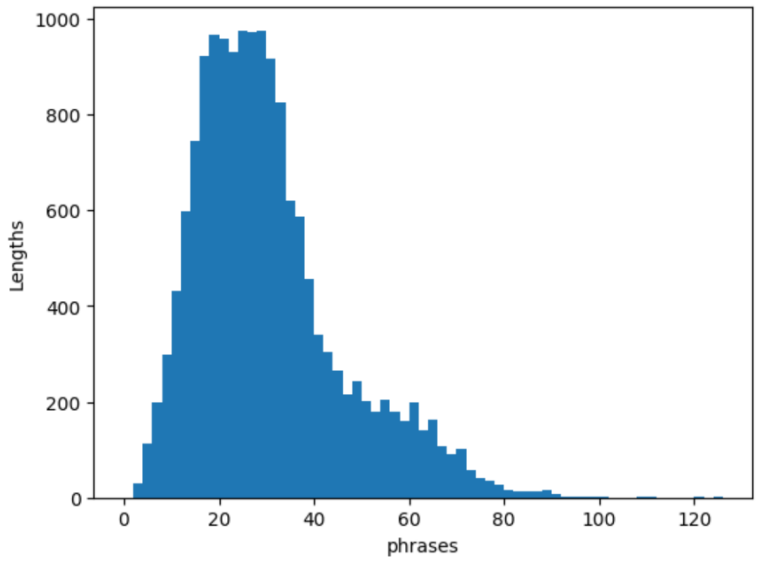

FIG. 3.14 : Longueurs des phrases d'entraînement

Longueurs des phrases a tester :

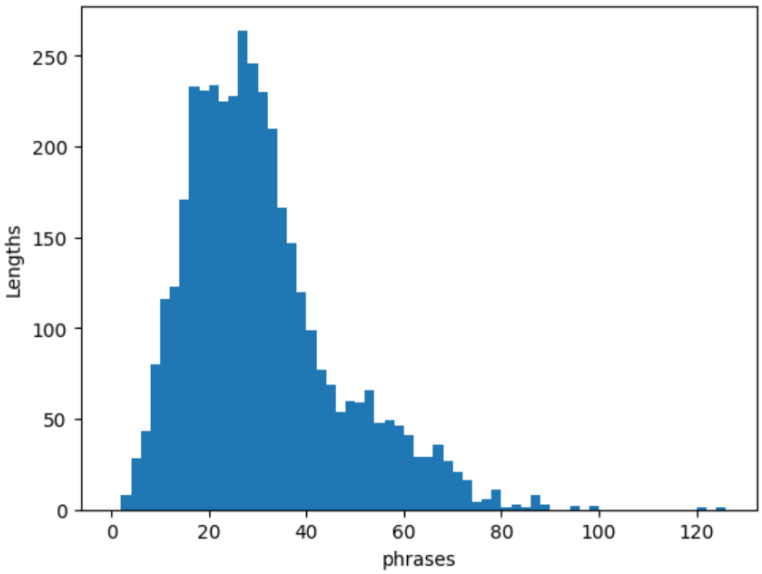

FIG. 3.15 : Longueurs des phrases a tester

```python
class ClassificationDataset(Dataset):
    def __init__(self, text, target, model_name, max_len, label_map):
      super(ClassificationDataset).__init__()

      self.text = text
      self.target = target
      self.tokenizer_name = model_name
      self.tokenizer = AutoTokenizer.from_pretrained(model_name)
      self.max_len = max_len
      self.label_map = label_map


    def __len__(self):
      return len(self.text)

    def __getitem__(self,item):
      text = str(self.text[item])
      text = " ".join(text.split())

      inputs = self.tokenizer(
          text,
          max_length=self.max_len,
          padding='max_length',
          truncation=True
      )
      return InputFeatures(**inputs,label=self.label_map[self.target[item]])
```

FIG. 3.16 : Partie du code montrant le processus de classification des datasets.

L'interface graphique de l'application, à travers laquelle nous affichons et analysons les résultats, il contient de nombreux onglets comme requis.

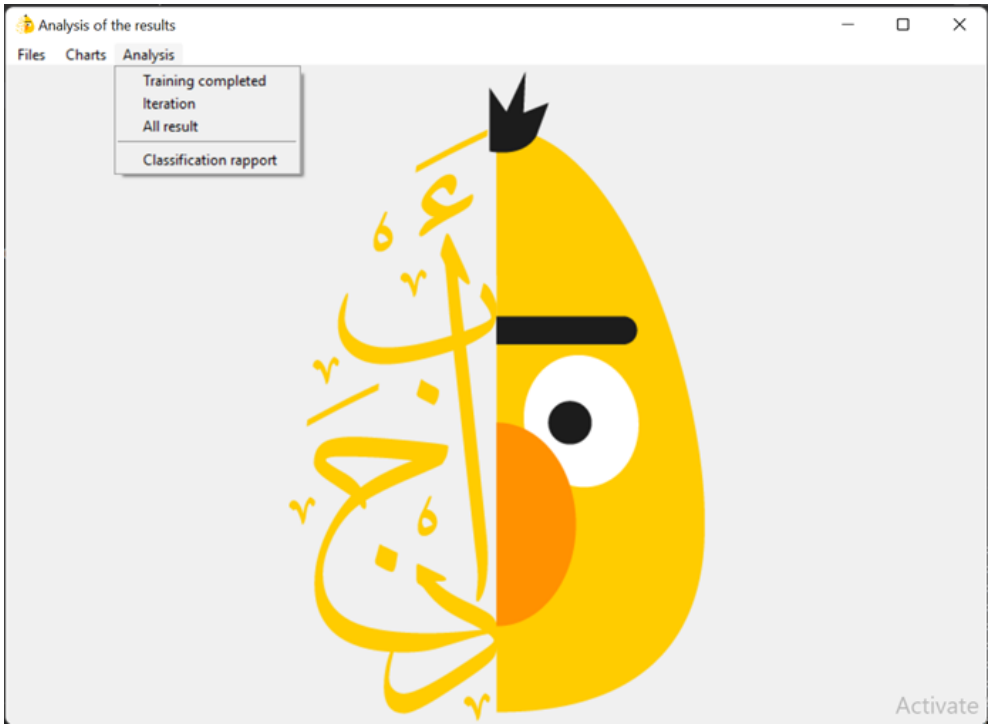

FIG. 3.17 : interface graphique De l'application

Résultats

L'arabe est une langue riche en vocabulaire avec relativement peu de ressources et une structure moins explorée par rapport à l'anglais et au français en raison de sa difficulté et de la rareté des processeurs disponibles. En raison de ces limitations, les tâches de traitement de la langue arabe naturelle (TAL) telles que l'analyse des sentiments (SA) se sont avérées très difficiles à aborder. Récemment, avec l'augmentation des modèles basés sur des adaptateurs, les modèles basés sur AraBERT spécifiques au langage se sont avérés très efficaces pour comprendre le langage, à condition qu'ils soient pré-formés sur un très grand ensemble. Ces modèles ont pu établir de nouvelles normes et obtenir des résultats de pointe pour la plupart des tâches TAL. Nous avons formé le modèle AraBERT sur différents ensembles de données pour produire un nouveau modèle qui a bien fonctionné sur la plupart des tâches de TAl en arabe testées.

Dans cette partie, nous allons présenter dans le tableau ci-dessous les résultats les plus importants obtenus grâce au modèle araBERT, tout en conservant les mêmes paramètres du modèle de classification, ainsi que les mêmes opérations de prétraitement :

	Ara-BERT
Précision d'entraînement	0.98
taux d'erreur d'entraînement	0.02
Précision de test	0.99
taux d'erreur de test	0.02
Temps d'entrainement	25 min

Tab. 3.2 : Les résultats des valeurs statistiques du processus de mise en œuvre du modèle

Rapport de classification

	precision	recall	f1-score	support
Mixed	0.42	0.02	0.04	226
Negative	0.80	0.89	0.84	1443
Neutral	0.83	0.85	0.84	1408
Positive	0.73	0.75	0.74	903
accuracy			0.79	3980
macro avg	0.69	0.63	0.61	3980
weighted avg	0.77	0.79	0.77	3980

Fig. 3.18 : Rapport de classification

Precision, Recall and F-score

À titre d'illustration, nous considérons le cadre simple suivant : chaque objet est associé à une étiquette binaire l qui rend compte de l'exactitude de l'objet par rapport à la tâche à accomplir. En outre, le système produit une affectation z indiquant s'il croit que l'objet est correct (ou pertinent) ou non. Le résultat expérimental peut être résumé de manière pratique dans un tableau de confusion :

		Assignment z	
		+	-
Label ℓ	+	TP	FN
	-	FP	TN

Fig. 3.19 : tableau de confusion

où + et - signifient pertinent et non pertinent, TP (resp. TN) signifie vrai positif (resp. négatif) et FP (resp. FN) signifie faux positif (resp. négatif). À partir de ces chiffres, on peut calculer la précision (p) et le rappel (r) :

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \tag{3.1}$$

En prenant la moyenne harmonique (pondérée) de la précision et du rappel, on obtient le F-score :

$$F_\beta = (1 + \beta^2)\frac{pr}{r + \beta^2 p} = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2 FN + FP} \tag{3.2}$$

Après avoir vérifié les meilleurs hyperparamètres, nous utilisons la section de formation normale et réentraînons le modèle en utilisant les paramètres dont nous disposons.
Soit on regroupe les modèles ensemble.
Afin de suivre de plus près les résultats obtenus, il devient plus clair de les suivre à travers des graphiques. Les graphiques suivants nous permettent de suivre l'évolution de la précision et du taux d'erreur lors de l'utilisation les version araBERT.

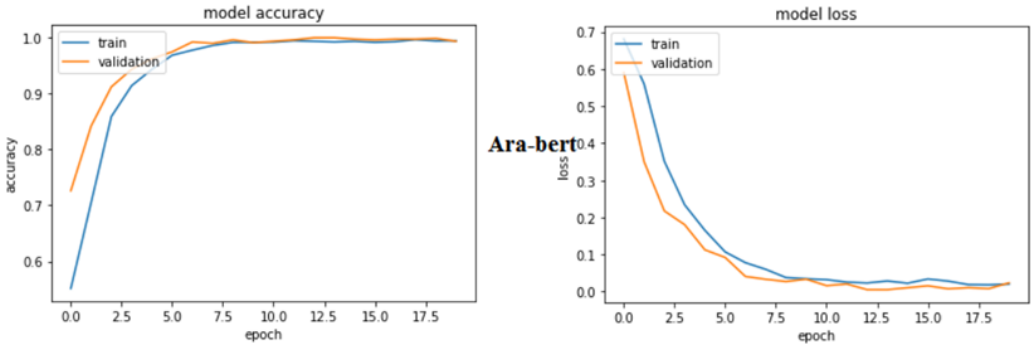

FIG. 3.20 : Évolution de la précision et du taux d'erreur AraBERT

3.7 Conclusion

Dans ce chapitre, nous avons présenté comment effectuer les différentes étapes de création d'un modèle de classification et d'analyse des sentiments, comme nous l'avons précédemment expliqué en détail la conception de ces étapes.
Nous avons également créé une application de model araBERT , où nous constatons que la version qui a été pré-formée en langue arabe a obtenu des résultats plus fluides et plus efficaces.

Conclusion et perspectives

Conclusion générale

Dans ce travail, nous explorons le domaine de l'analyse des sentiments, qui, comme tous les autres domaines du traitement du langage naturel, a connu une évolution majeure depuis les années 2000, et a connu une évolution et un intérêt significatifs depuis sa création. l'apprentissage en profondeur. Pour obtenir ces résultats, nous avons passé un temps considérable à lire et à réviser des publications et des articles pour comprendre ces concepts et comment appliquer des modèles d'apprentissage en profondeur à notre problème.

Pour effectuer une analyse de sentiment sur un texte arabe, nous avons proposé une méthode basée sur le deep learning et plus précisément sur le modèle AraBERT, ou nous avons entraîné notre modèle d'analyse de sentiment sur le jeu de données sur ce modèle et avons conclu des résultats que ce modèle est puissant et efficace dans le traitement du langage arabe.

Enfin, avant de passer aux perspectives, ce travail nous a permis de compléter notre travail en connaissant et en enrichissant le transformateur, et surtout de faire le premier pas vers l'apprentissage en profondeur, qui est l'un des domaines les plus importants de l'intelligence artificielle. Pour les opinions et les travaux futurs susceptibles d'améliorer notre système l'évaluation des sentiments, nous avons des idées telles que :

- Détection et suppression des mots indésirables.
- Tester notre modèle sur d'autres données

Bibliographie

[1] Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv :2001.09977*, 2020.

[2] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert : Transformer-based model for arabic language understanding. *arXiv preprint arXiv :2003.00104*, 2020.

[3] U. Bhattacharya B. Chaudhuri. Efficient training and improved performance of multilayer perceptron in pattern classification. *Neurocomputing 34(1)*, page 11–27, 2000.

[4] Mohamed Bouaziz. *Réseaux de neurones récurrents pour la classification de séquences dans des flux audiovisuels parallèles*. PhD thesis, Université d'Avignon, 2017.

[5] P. Charaudeau. Les conditions d'une typologie des genres télévisuels d'information. *Réseaux 15(81)*, pages 79–101.

[6] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv :1911.02116*, 2019.

[7] R. J. Williams D. E. Rumelhart, G. E. Hinton. Learning internal representations by error propagation. *Rapport technique, DTIC Document*, pages 79–211, 1985.

[8] Jean-Yves Antoine Damien Nouvel, Nathalie Friburger. Traitement Automatique des Langues pour les Systèmes d'Information. *support du cours pour Master SIAD–M2, Université FRANCOIS–RABELAIS*.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*, 2018.

[10] Fouad Soufiane Douzidia. Résumé automatique de texte arabe. *Mémoire présenté à la Faculté des études supérieures en vue de l'obtention du grade de M.Sc en informatique, Université de Montréal*, Septembre, 2004.

[11] J. L. Elman. Finding structure in time. *Cognitive science 14(2)*, pages 79–211, 1990.

[12] Claire Gardent. Traitement des Langues Naturelles (TAL). Septembre 2011 , ENS Cachan.

[13] F. Gers. Long short-term memory in recurrent neural networks. *Thèse de Doctorat, Universität Hannover*, 2001.

[14] A. Graves. Supervised Sequence Labelling with Recurrent Neural Networks. *Thèse de Doctorat, Citeseer*, 2008.

[15] F. Beaufays H. Sak, A. Senior. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognitio. *arXiv preprint arXiv :1402.1128*, 2014.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[17] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv :1801.06146*, 2018.

[18] M. Lapata J. Cheng, L. Dong. Long short-term memorynetworks for machine reading. *arXiv preprint arXiv :1601.06733*, 2016.

[19] H. White K. Hornik, M. Stinchcombe. Multilayer feedforward networks are universal approximators. *Neural networks 2(5)*, pages 359–366, 1989.

[20] Y. Zhang D. Yu G. Zweig Y. Shi K. Yao, B. Peng. Spoken language understanding using long short-term memory neural networks. *Dans les actes de Spoken Language Technology Workshop (SLT), 2014 IEEE, 189–194. IEEE.*, 2014.

[21] T. Fischer C. Krauß. Deep learning with long short-term memory networks for financial market predictions. *FAU Discussion Papers in Economics.*, 2017.

[22] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32, 2019.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[24] M. Schuster K. K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on 45(11)*, pages 2673–2681, 1997.

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

[26] D. Li J. Qian. Text sentiment analysis based on long short-term memory. *International Conference on ComputerCommunication and the Internet (ICCCI)*, pages 471–475, 1997,IEEE.

[27] Adam Roberts, Colin Raffel, Katherine Lee, Michael Matena, Noam Shazeer, Peter J Liu, Sharan Narang, Wei Li, and Yanqi Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019.

[28] P. Frasconi J. Schmidhuber S. Hochreiter, Y. Bengio. Gradient flow in recurrent nets : the difficulty of learning long-term dependencies. 2001.

[29] D. Chai S. L. Phung, A. Bouzerdoum. Skin segmentation using color pixel classification : analysis and comparison. *IEEE transactions on pattern analysis and machine intelligence 27(1)*, page 148–154, 2005.

[30] G. Sabah. I, l'intelligence artificielle et le langage. *Hermès, Paris*, 1998.

[31] A. Graves J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. *. Dans les actes de International Joint Conference on Neural Networks (IJCNN), Volume 4, 2047–2052. IEEE*, 2005b.

[32] S. Hochreiter J. Schmidhuber. Long short-term memory. *Neural computation 9(8)*, 1997.

[33] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[34] Benoît TROUVILLIEZ. Traitement Automatique des Langues (TAL). *Intelligence Artificielle (IA), Analyse sémantique et Clusterings*, 31 mars 2010.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell : A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[37] J. Véronis. Informatique et linguistique, support du cour pour licence en sciences du langage. *Centre Informatique pour les Lettres et Sciences Humaines, Université de Provence, France*, 2001.

[38] Y. Wang H. Yu Y. Wang X. Ma, Z. Tao. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C : Emerging Technologies 54*, pages 187–197, 2015.

[39] T.-L. Wu S.-K. Jeng J.-H. Chen Y.-P. Lin, C.-H. Wang. Multilayer perceptron for eeg signal classification during listening to emotional musi. *Dans les actes de TENCON Region 10 Conference*, page 1–3, 2007 ,IEEE.

[40] François Yvon. Une petite introduction au traitement automatique des langues naturelles. *Support du cour*.