N°d'ordre: .....

République Algerienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la Recherche scientifique

Université Djillali Liabès de Sidi Bel Abbès
Faculté des Sciences Exactes
Département d'informatique

Filière : Informatique
Spécialité : Web et Ingenierie des Connaissances (WIC)

Mémoire de Master

Hate Speech Detection on Arabic Social Media using Deep Learning

Par

Mlle Charaf Belfekroun

Mémoire soutenu devant le jury composé de :

Dr.	Mohamed Benhammouda	UDL SBA	(Président)
Dr.	Djamil Mehadji	UDL SBA	(Membre)
Pr.	Zakaria Elberrichi	UDL SBA	(Encadrant)

Année Universitaire : 2020 - 2021

Je dédie ce mémoire,
A mes parents, pour l'éducation qu'ils m'ont prodigué, pour le sens du devoir qu'ils m'ont enseigné depuis toujours
A mon frère Mohamed Ilies, mes soeurs Farah et Manel
A mes grands-parents que dieu les bénisse
A mes tantes, mes cousins et cousines
A tous mes collègues de la promotion
Mes enseignants
Mes proches
Mes amis
Tous ceux qui m'ont encouragé et soutenu

Remerciements

Ce mémoire est le résultat d'un travail riche en efforts, en sacrifices, en acharnement, et en recherche. En préambule, je tiens à remercier ALLAH le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail, je voudrais également adresser tous mes remerciements aux personnes avec lesquelles j'ai pu échanger et qui m'ont aidé pour la réalisation de ce projet qui me tien vraiment à coeur.

En commençant par remercier tout d'abord mon encadreur Monsieur le **Pr Z.Elberrichi** , pour son aide précieuse, ses conseils judicieux et pour le temps qu'il m'a consacré pour mener à bien ce projet.
Mes vifs remerciements vont également aux membres du **jury** pour l'intérêt qu'ils ont porté à mon projet de fin d'études en acceptant d'examiner mon travail et de l'enrichir par leurs propositions et critiques constructives.

Mes remerciements sincères vont aussi à l'ensemble des **Enseignants et des professeurs** qui m'ont accompagné durant mon cursus universitaire.
Enfin, je voudrais adresser mes plus sincères remerciements à ma famille : **mes parents, mon frère** et **mes soeurs** ,**mes petits neveux**, et à tous mes proches et amis, qui m' ont accompagné, aidé, soutenu et encouragé tout au long de la réalisation de ce projet.

Résumé

Les réseaux de neurones représentent la nouvelle génération des algorithme d'apprentissage .Dans ce mémoire ,nous avons utilisé les réseaux de neurones pour la classification du texte arabe , plus précisément pour la détection des discours haineux sur les réseaux sociaux .

Pour cela nous avons utilisé deux modèles .Le premier est un modèle hybride CNN-BiLSTM et le deuxième est un BiLSTM seulement , ainsi qu'une représentation de mot avec AraVec.Plusieurs expérimentations ont été effectuées afin d'ajuster les paramètres par rapport à notre problème et d'avoir de meilleur résultats .

Les meilleurs résultats que nous avons obtenus sont 0.87 de Rappel et F-mesure ,acuuracy à 0.87 sur les commentaires de "YouTube" du dataset de Alakrot et al. [2018] et cela avec le modèle BiLSTM ;et aussi 0.8429 d'accuracy pour le dataset qui contient que les "Tweets" ; et au final nous avons obtenu 0.81 d'accuracy pour le premier dataset Mubarak et al. [2017] qui contient des commentaires de différentes plateformes "Facebook ,Twitter,Youtube" .

Mots-Clés : Réseaux de neurones,Deep Learning ,Hate Speech,CNN,BiLSTM ,Texte arabe ,Classification de texte.

Table des matières

Liste des figures

Liste des tableaux

Nomenclature

IA : Intelligence artificielle
ML : Machine Learning
DL : Deep Learning
MAJ : Mise A Jour
ANN : Artificiel Neural Network
CNN : Convolutional Neural Networks
LSTM : Long Short Term Memory
BiLSTM : Bi-directional LSTM
DNN : Deep Neural Network
DBN : Deep Belief Network
RAE : Recursive Auto Encoder
MLP : MultiLayer Perceptron
GRU : Gated Recurrent Units
KNN : K-Nearest Neighbors
SVM : Support Vector Machine
CBOW : Continuous Bag of Words
EGY : Dialecte égyptien
LEV : Dialecte levantin
GULF : Dialecte du golf
ATB : Arabic TreeBanks
lr : Learning Rate
TP : True Positive
TN : True Negative
FN : False Negative
FP : False Positive
F-mesure :F1-score
Acc : Accuracy
LDC : Linguistic Data Consortium
MADAMIRA : Morphological Analysis and Disambiguation of Arabic
ArSAS : An Arabic Speech-Act and Sentiment Corpus of Tweets
L-HSAB : Levantine Twitter Dataset for Hate Speech and Abusive Language

Introduction 1

Les réseaux sociaux sont un moyen de communication populaire caractérisé par la capacité d'atteindre un large public au sein d'un petit cadre temporel.

Cela permet également l'anonymat il est moins réglementé que d'autres formes de médias. Cela rend "social médias" facile à exploiter pour diffuser des discours et commentaires haineux et offensants sur des individus et des groupes Izsak-Ndiaye [2015].

Les commentaires offensifs tels que les insultes, les humiliations, les menaces et les attaques peuvent provoquer une détresse émotionnelle et affecter la santé des utilisateurs des médias sociaux Mengü et Mengü [2015].
Hate Speech sur les réseaux sociaux a prouvé que les médias incitent aux crimes haineux et encouragent les stéréotypes négatifs et les préjugés contre les groupes minoritaires Gelashvili [2018] .

Le fait de laisser le discours de haine ,non réglementé sur les réseaux sociaux fait valoir que le Hate Speech fait normalement partie de la société,ce qui encourage l'expression du discours de haine en dehors des médias sociaux également .

Le rapport de l'ONU Izsak-Ndiaye [2015] sur Minority Issues insiste pour que plus de mesures soient prises, surveiller les discours de haine sur les réseaux sociaux et les supprimer le plus rapidement possible . C'est parce que plus cela prend de temps pour que les discours de haine soient détectés et supprimés, plus il y a d'utilisateurs qui sont exposés et affectés par eux .

La détection automatique d'un langage offensant sur les réseaux sociaux permettrait une suppression rapide de ce contenu, avant qu'il n'atteigne et nuit à tous les utilisateurs de médias sociaux.

L'utilisation de la langue arabe dans les réseaux sociaux est répandue et en constante augmentation.

Depuis 2017,le rapport arabe sur les médias sociaux estime que les utilisateurs de Facebook de la région arabe représentent 8,4% de tous les utilisateurs de Facebook ce qui représente plus de 150 millions d'utilisateurs arabes Salem [2017]. La réglementation du contenu des médias sociaux en arabe est donc un problème important.

La détection automatique du hate speech et du langage abusif sur les réseaux sociaux arabes permettra à ces réglementations d'être mises en œuvre.

Background 2

2.1 Hate Speech (discours de haine)

Un discours de haine (ou « discours haineux », « discours de la haine »), désigne un type de discours ou de système qui (au-delà de la violence ou de l'injure ponctuelle en termes de force et de nature1) attaque une personne ou un groupe de personnes sur la base de caractéristiques diverses (couleur de peau, ethnie, âge, sexe, orientation sexuelle, religion, etc.). [1]
L'Histoire a montré que les discours haineux peuvent conduire à des suicides, lynchages, fusillades de masse attaques par explosifs, guerres, crimes de masses et processus génocidaires.

2.2 Réseau social

Dans le domaine des technologies, un réseau social consiste en un service permettant de regrouper diverses personnes afin de créer un échange sur un sujet particulier ou non. En quelque sorte, le réseau social trouve ses origines dans les forums, groupes de discussion et salons de chat introduits dès les premières heures d'Internet.
Depuis le début des années 2000, la présence des réseaux sociaux, également appelés réseaux communautaires, devient de plus en plus importante et tend à se multiplier selon diverses caractéristiques [2].
Les premiers réseaux sociaux de grande envergure (MySpace et Facebook) se sont positionnés en tant que services généralistes sur lesquels chacun peut partager le contenu de son choix, quel qu'en soit le sujet, avec ses contacts.
Notons que l'avènement des smartphones a transformé les réseaux sociaux pour en revoir leurs usages ou les confiner uniquement au téléphone.

2.3 Data mining

L'ensemble du processus de Data Mining comprend trois phases principales :

1. Prétraitement des données - Le nettoyage, la sélection et la transformation des données.

1. https://bit.ly/3cyB2nC
2. https://bit.ly/3pInVW3

2. Extraction de données .
3. Évaluation et présentation des données - Analyse et présentation des résultats.

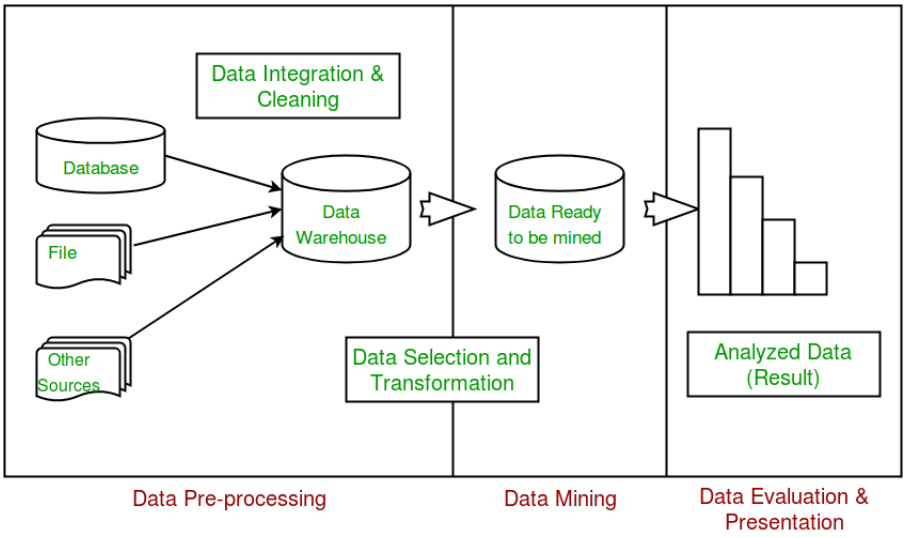


FIGURE 2.1 – *Processus de Data Mining*

2.3.1 Text mining

- Texte mining ≈ Analyse de texte
- Transformez les données textuelles en informations de haute qualité ou connaissance Actionnable.
  1. Minimise l'effort humain (consommation de données texte)
  2. Fournit des connaissances pour une prise de décision optimale
- Relatif à l'extraction du texte, qui est une composante essentielle de tout système texte mining. L'extraction de texte peut être une étape préprocesseur pour le texte mining

2.3.2 Opinion mining

Opinion mining (aussi appelé sentiment analysis) est l'analyse des sentiments à partir de sources textuelles dématérialisées sur de grandes quantités de données (big data).
L'objectif de l'opinion mining est d'analyser une grande quantité de données afin d'en déduire les différents sentiments qui y sont exprimés. Les sentiments extraits peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté.

2.4 LE TEXTE ARABE

La langue arabe est l'une des langues les plus répandues,c'est la 5^{ieme} langue la plus utilisée au monde et la 5^{ieme} langue la plus utilisée sur Internet.

C'est l'un des langages les plus difficiles au monde avec sa riche morphologie, sa syntaxe complexe et sa sémantique difficile. Cela rend son analyse et son traitement automatique très difficiles et complexes.

2.4.1 Particularité du texte arabe

- La langue arabe compte 28 lettres qui peuvent se raccorder entre elles (sauf les lettres ا ,و ,ز ,ر ,د qui ne se joignent pas à gauche) et qui changent de forme et de présentation selon leurs positions (au début, au milieu ou à la fin du mot).
- La langue arabe a une manière spéciale d'écrire ;contrairement à l'anglais, au français et à toutes les langues occidentales en général, la langue arabe s'écrit de droite à gauche
- Il n'y a pas de minuscules et de majuscules pour les lettres arabes.

2.5 Deep Learning

Le Deep Learning est un ensemble de techniques d'apprentissage automatique qui a permis des avancées importantes en intelligence artificielle dans les dernières années.
L'apprentissage profond est basé sur ce qui a été appelé, par analogie, des « réseaux de neurones artificiels »,composés de milliers d'unités (neurones) qui effectuent chacune de petites opérations simples. Les résultats d'une première couche de neurones servent d'entrée aux calculs d'une deuxième couche et ainsi de suite.

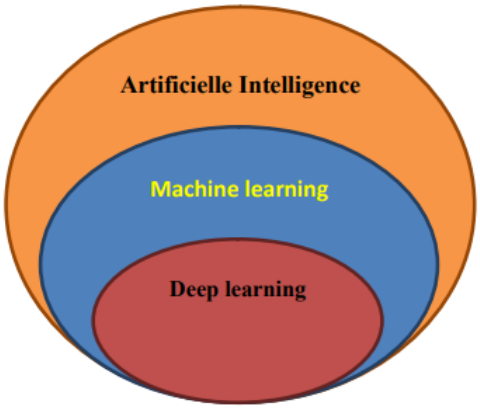


Figure 2.2 – *La relation entre l'IA, le ML et le DL*

2.5.1 Deep Learning vs Machine Learning

Une des grandes différences entre le DL et les algorithmes de ML traditionnels c'est qu'il s'adapte bien, plus la quantité de données fournies est grande plus les performances d'un algorithme de Deep Learning sont meilleures.
Autre différence entre les algorithmes de ML traditionnelles et les algorithmes de Deep Learning c'est l'étape de l'extraction de caractéristiques. Dans les algorithmes de ML traditionnelles l'extraction de caractéristiques est faite manuellement, c'est une étape difficile et coûteuse en temps et requiert un spécialiste en la matière alors qu'en Deep Learning cette étape est exécutée automatiquement par l'algorithme.
L'apprentissage en profondeur se différencie des approches classiques principalement par l'ensemble de modèles puissants qu'elle focalise sur eux . Ces modèles consistent en de nombreuses transformations successives des données qui sont enchaînées de **haut en bas**, d'**où le nom** d'apprentissage en profondeur.

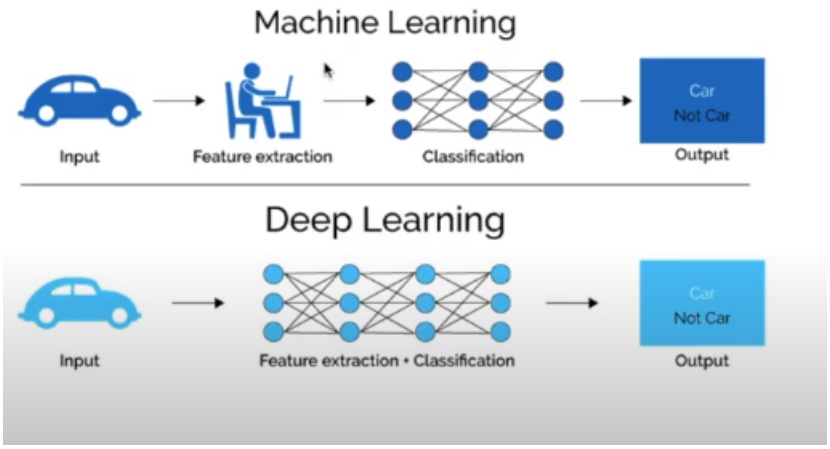


FIGURE 2.3 – *Le procédé du ML comparé à celui de DL*

2.6 ARCHITECTURE DES RÉSEAUX DE NEURONES ARTIFICIEL

2.6.1 Artificiel neural network (ANN)

Il existe un grand nombre de variantes d'architectures profondes. La plupart d'entre elles sont dérivées de certaines architectures parentales originales .
Il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données.

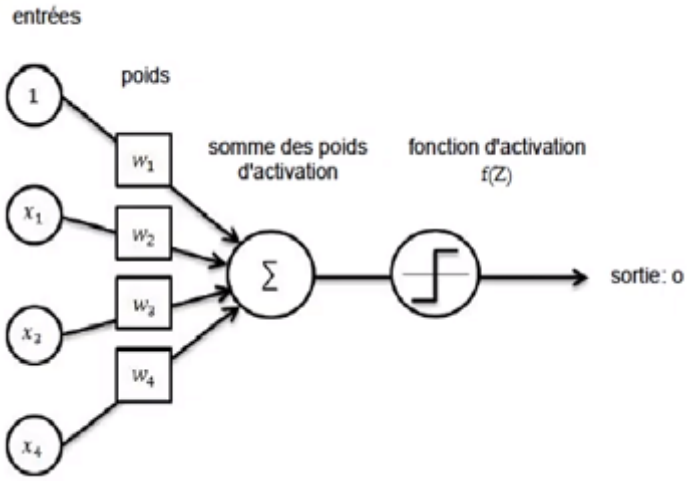


FIGURE 2.4 – *Représentation d'un réseau de neurone*

Par exemple, pour donner un prêt a un client il y a des caractéristiques importantes que le modèle doit prendre en considération telles que : le salaire , âge , il travail ou pas,il a une retraite ou pas ...

L'architecture du réseau de neurones se compose :
— La couche d'entrée qui comprend les caractéristiques telles que l'âge, le salaire annuel, le solde bancaire, etc.
— La couche de sortie qui génère la prédiction du modèle qui, dans votre cas, est de savoir si un client devrait obtenir un prêt ou non.
— Les couches en dehors des couches d'entrée et de sortie sont appelées couches cachées.

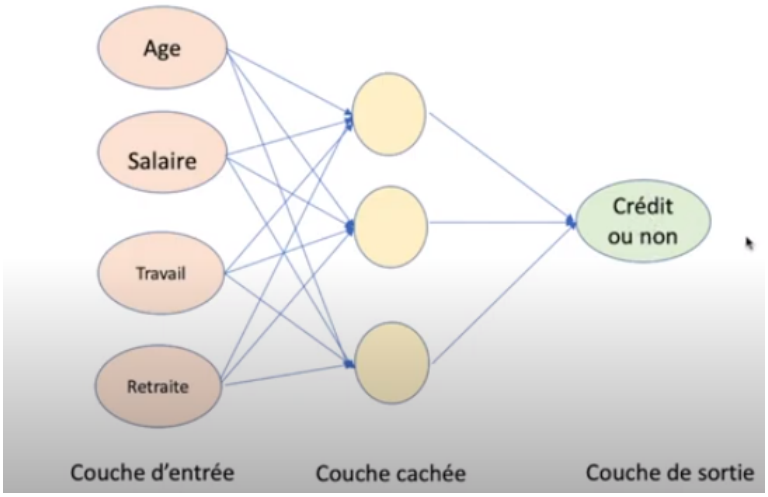


FIGURE 2.5 – *Exemple ANN*

Le réseau effectue une propagation directe.
Pour simplifier, considérons seulement deux caractéristiques en entrée, a savoir l'âge et le statut de retraite, le statut de travail étant un nombre binaire (1- travail et 0 — sans travail) sur la base duquel vous ferez des prédictions.

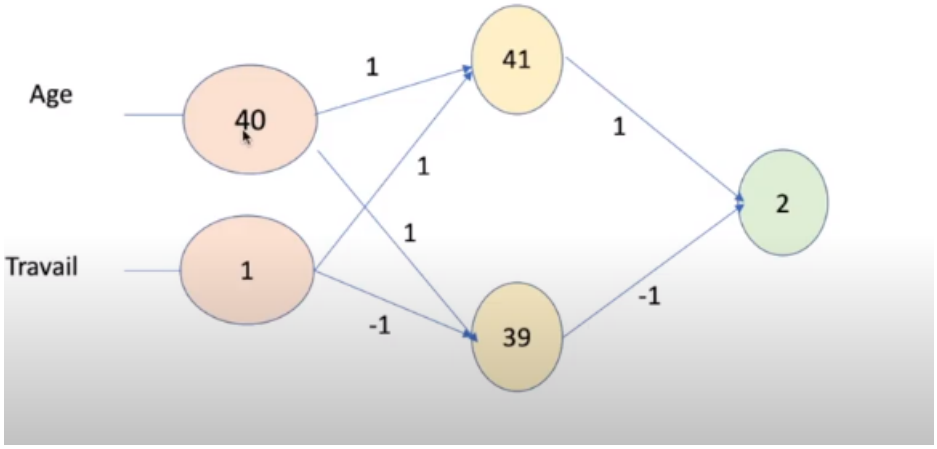


FIGURE 2.6 – *Explication de l'exemple ANN*

Pour expliquer ; on a un client qui a 40 ans et qui travaille on attribut les poids et on calcule :
$40 * 1 + 1 * 1 = 41$.
$40 * 1 + 1 * (-1) = 39$
En suite : $41 * 1 + 39 * (-1) = 2$ on obtient 2 comme résultat d'où ça nécessite l'utilisation d'une **fonction d'activation**.

Fonction d'activation :

Le processus de multiplication-addition n'est que la moitié du fonctionnement d'un réseau de neurones ; il y a plus !
Pour utiliser la puissance prédictive maximale, un réseau de neurones utilise une fonction d'activation.
Une fonction d'activation permet au réseau neuronal de capturer les non-linéarités présentes dans les données.

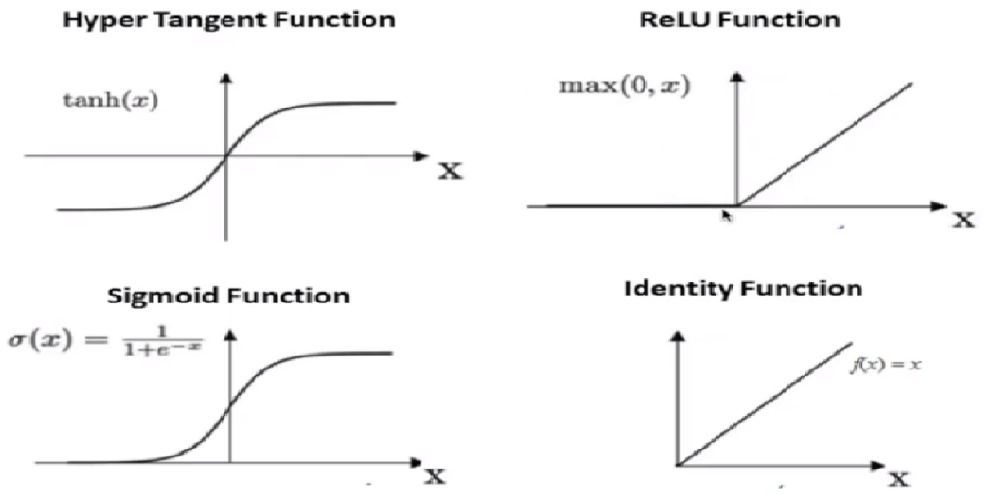


FIGURE 2.7 – *Fonctions d'activation*

Les poids (paramètres) :

Les poids sont la clé du deep learning que nous entraînons et qu'on mets a jour lorsque nous adaptons un réseau neuronal aux données.
Ces poids peuvent être appelés paramétrés également.

Fonction de coût

C'est une fonction qui prend en entrée la valeur prédite et la valeur réelle (évaluation) et calcule l'erreur (il y a différent algorithme) pour voir a quel point on est précis .
On l'utilise pour faire la MAJ des poids (rétro propagation) et pour avoir plus de précision .

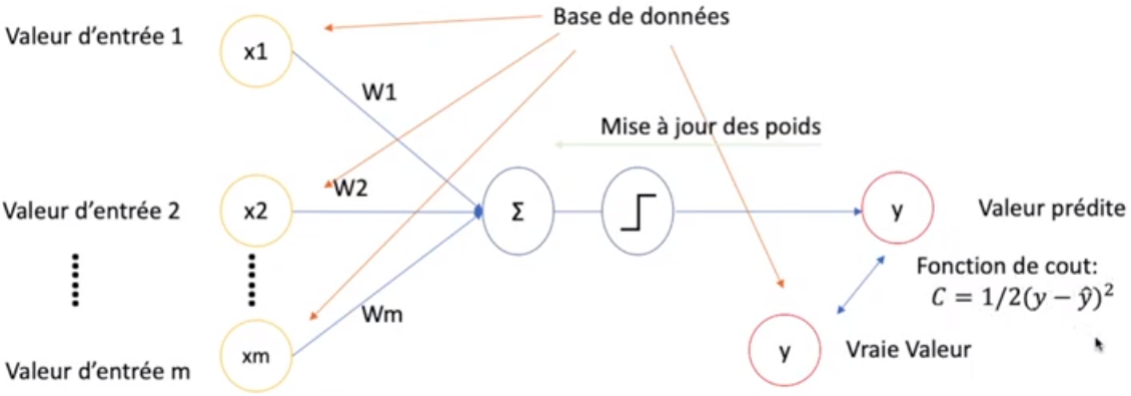


FIGURE 2.8 – *Fonctionnement de MAJ des poids et cout*

2.6.2 Convolutif Neural Network(CNN)

Un réseau de neurones convolutifs (abrégé en CNN) est un algorithme de deep learning (apprentissage profond en français) doué dans l'analyse d'images. En effet, il s'appuie sur des filtres, dans le style de nos logiciels de retouche photo, par exemple le floutage ou la mise en avant des contours, pour extraire les informations de l'image qu'il juge "pertinentes" (typiquement la présence d'une roue de voiture, une forte proportion de lignes horizontales dans telle partie de l'image, etc...).
Ici, on parle de convolutions 2D (i.e. à 2 dimensions) car, pour chaque pixel de l'image, on va regarder ce qu'il y a autour (en carré). A noter qu'une convolution est l'application d'un filtre à l'image [3].

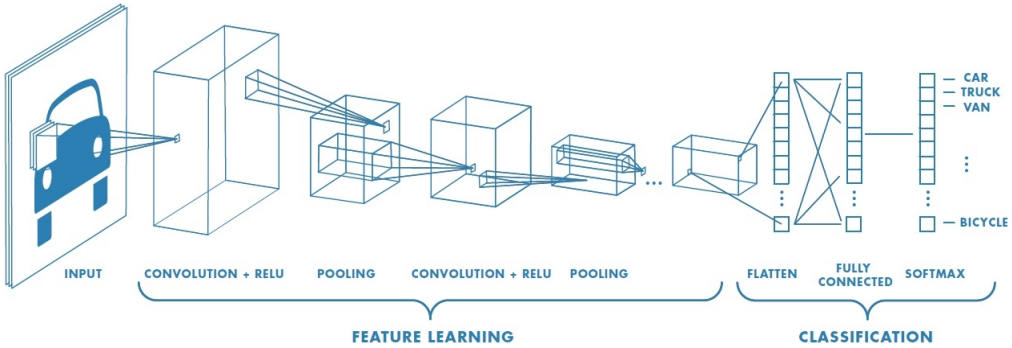


FIGURE 2.9 – *Exemple de CNN : l'image à gauche est analysée par le réseau dans sa phase "feature learning" qui renvoie alors son "résumé/analyse" au réseau suivant*

Cependant, les réseaux de neurones convolutifs sont également très utilisés pour analyser des séries temporelles avec des convolutions 1D car le principe est similaire à celui d'une image : la donnée à l'instant t est liée à la donnée à l'instant t-1 et à t+1 (comme un pixel est lié à ceux qui l'entourent), avec un seul "axe" possible ici (i.e. on n'a qu'une dimension spatiale = temporelle). C'est pourquoi on les retrouve en traitement du langage naturel (NLP en anglais) où la notion de temps est appliquée à l'enchaînement des mots dans une phrase.

2.6.3 Recurrent neural network (RNN)

Les couches de réseaux neuronaux récurrentes (RNN : Recurrent Neural Networks) sont des entités primitives qui permettent aux réseaux neuronaux d'apprendre à partir de séquences d'entrées
Si la séquence que nous traitons est une phrase de 3 termes par exemple, le réseau va être déroulé en un réseau neuronal à 3 couches, une couche pour chaque mot, la figure suivante représente cette idée :

3. https://bit.ly/2R6lZdl

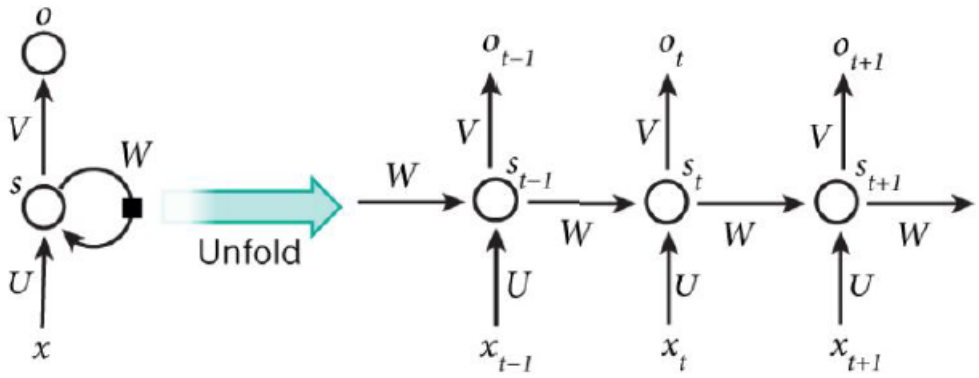


FIGURE 2.10 – *Un exemple d'un réseau récurrent qui se déroule*

Les formules qui dirigent le calcul sont les suivantes :

- x_t est l'entrée au temps t
- s_t est l'état caché à l'instant t
  s_t est calculé en fonction de l'état caché précédent et de l'entrée à l'étape courante avec :
  $x_t = f(u_{xt} + w_{st-1})$
- La fonction f est la fonction d'activation
- O_t à l'instant . Par exemple, si nous souhaitons prédire le mot suivant dans une phrase, ce serait un vecteur de probabilités à travers notre vocabulaire, nous utiliserons alors la fonction d'activation softmax avec :$o_t = softmax(v_{st})$.

2.6.4 Long Short Term Memory(LSTM)

Un "Long Short Term Memory" (LSTM) est un algorithme de deep learning issu des réseaux de neurones récurrents (RNN). Ce type de réseau dispose d'un mécanisme de mémoire qui lui permet de retenir des informations d'une prédiction à l'autre : il est parfaitement adapté au traitement du langage naturel puisqu'à mesure que l'on lit une phrase, il faut en retenir les éléments clés (le sujet par exemple).

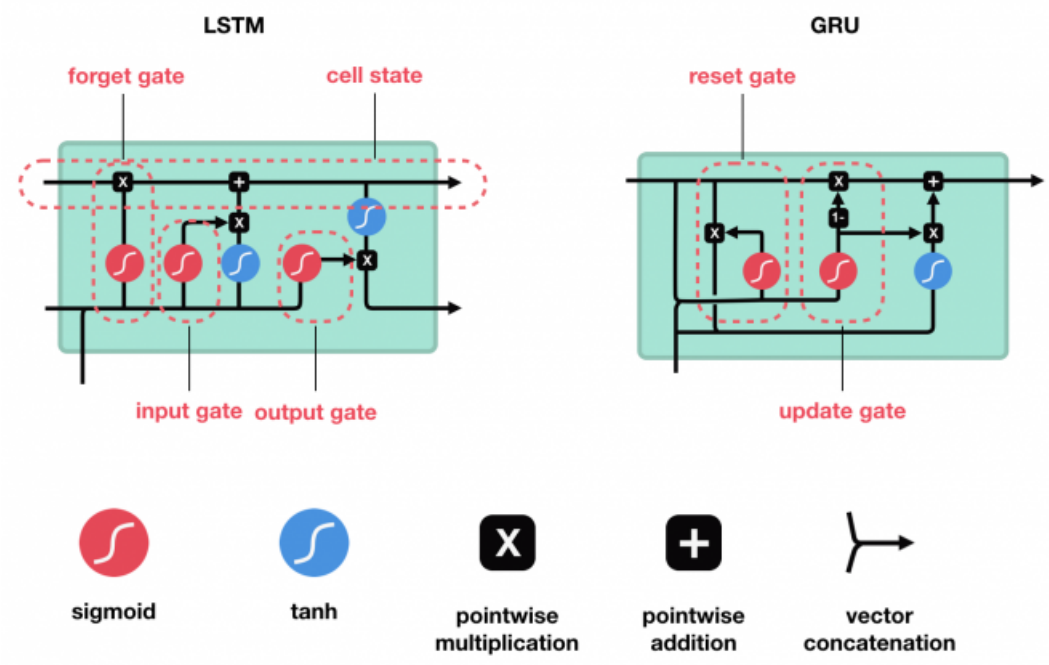


FIGURE 2.11 – *La structure interne d'un LSTM et d'un GRU*

Remarque : l'architecture des LSTM est proche de celle d'un GRU (Gated Recurrent Unit), donc il est fréquent de voir utilisé l'un ou l'autre. A noter aussi qu'une version dite "bi-LSTM" existe. En résumé, son principe est de pouvoir prédire des deux côtés d'un mot (par exemple) : pour la phrase "Je * promener le *" le bi-LSTM sera capable de prédire la première et la dernière étoile en n'ayant reçu que le mot "promener" (bi-LSTM est la contraction de LSTM bidirectionnel) [4].

2.6.5 Gated Recurrent Unit(GRU)

Un réseau Gated Recurrent Unit (GRU), en français réseau récurrent à portes ou plus explicitement réseau de neurones récurrents à portes, est une variante des LSTM introduite en 2014. Les réseaux GRU ont des performances comparables aux LSTM pour la prédiction de séries temporelles (ex : partitions musicales, données de parole). Une unité requiert moins de paramètres à apprendre qu'une unité LSTM. Un neurone n'est associé plus qu'à un état caché (plus de cell state) et les portes d'entrée et d'oubli de l'état caché sont fusionnées (update gate). La porte de sortie est remplacée par une porte de réinitialisation (reset gate) [5].

4. https://bit.ly/34uckQC
5. https://bit.ly/3i4Fxtq

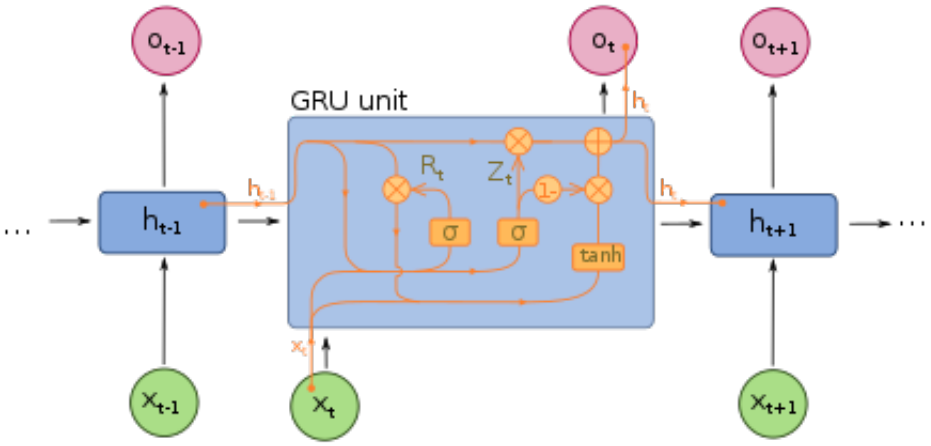


FIGURE 2.12 – *La structure interne d'un LSTM et d'un GRU*

Les réseaux neuronaux mentionnés, ne sont pas tous les réseaux qui existent

État de l'art 3

Un nombre limité de recherches sur le Hate Speech qui a été proposé pour la langue arabe , Abozinadah et al. [2015] est parmi les premiers qui s'intéressaient au Hate Speech en arabe en utilisant une approche de Machine Learning,et presque tous les travaux qui suivent utilisent cette approche .
De nos jour on a des machines puissantes par rapport à eux ,ce développement a donné chance aux techniques de Deep Learning pour prouver leur efficacité a améliorer les performances et qui est prouvé par la suite du chapitre .

3.1 Arabic Sentiment Analysis

Pour l'approche de Al Sallab et al. [2015] ont proposé d'utiliser plusieurs architecture des réseaux de neurones tel que DNN,DBN et RAE(Recursive Auto Encoder) ensuite ils ont comparé pour trouver le meilleure par leur études.

Au finale le RAE était reconnu étant meilleure pour ce cas d'étude après avoir fait :

- Une initialisation de l'arbre d'analyse avec les poids des mots ,word embedding .
- **Forward Path** Au début les noeuds parent sont initialisé à 0, pour le but de construire le meilleur arbre d'analyse avec un minimum d'erreur
- **Backward Path** Une fois l'arbe d'analyse construit les poids de RAE peuvent être ajustés.
- Ensuite ils répètent 2 à 3 fois, chaque cas .

Tout ce la est englobé dans la figure suivante .

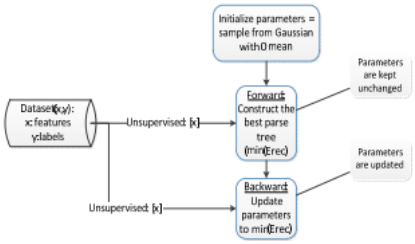


FIGURE 3.1 – *Procédure de formation de RAE avec découverte du meilleur arbre d'analyse .Al Sallab et al. [2015]*

L'architecture complète du systéme nécessite :
— Construction de la matrice word embedding
— Construire l'arbre d'analyse RAE et mis a jour des poids.
— Entraîner un classificateur en haut de la représentation pour ce cas c'est just Softmax layer .

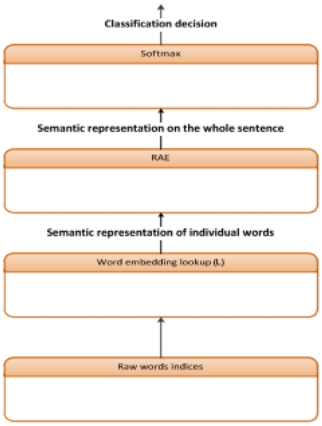


FIGURE 3.2 – *Classification des sentiments à l'aide de RAE .Al Sallab et al. [2015]*

LDC ATB dataset est utilisé , qui contient 944 instance pour l'apprentissage et 236 instances pour l'évaluation .
En appliquant l'architecture proposé sur le dataset ils aboutissent à 74,3% d'accuracy et 73,5% de F1-score.

Pour l'étude de Baly et al. [2017] ils ont pris 3 ensembles (EGY,LEV,GULF) chaque ensemble contient 610 tweet .
Ils analysent leurs données en tirant :
-Les sujets abordés fréquemment pour chaque ensemble
-Le langage, le style d'écriture de chaque ensemble .
Ils ont remarqué que la plupart des sujets abordés tourne autour des sujets de religions .

Prétraitement

Pour cette phase ils ont :

— Remplacé les mentions des twitteur et les URL par des tokens spéciales .
— Remplacé les emojis par leur signification en texte.
— Normalisé des données tout en enlevant les hashtags ,underscores et l'élongation de mot .
— Utilsé le Standard Arabic Morphological Analyzer(SAMA).
— Lemmatisé les données en utilisant MADAMIRA.

Opinion mining avec Recursive Neural Tensor Network

Chaque phrase est représentée sous la forme d'un arbre d'analyse binaire. Puis, à chaque nœud de l'arbre, une fonction de composition à base de tenseur combine les vecteurs des nœuds enfants (par exemple, C_1, C_2) et produit le vecteur du nœud parent (par exemple, P_1).
Ce processus se répète récursivement jusqu'à ce qu'il dérive un vecteur pour chaque nœud C_i dans l'arborescence, y compris le nœud racine qui correspond à la phrase entière. Ces vecteurs sont ensuite utilisés pour former un classificateur softmax pour prédire la distribution d'opinion $y^{C_i} \in R^K$ pour le texte représenté par le i^{ieme} nœud, où K est le nombre de classes d'opinion.

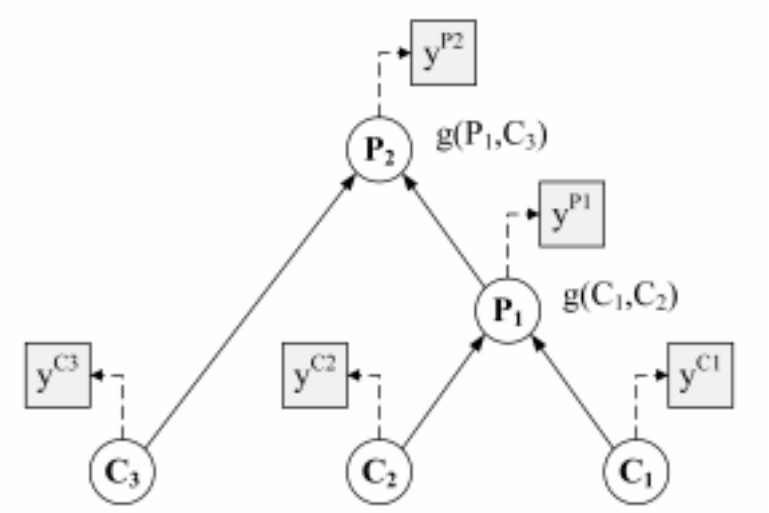


FIGURE 3.3 – *Applicaion de RNTN pour opinion mining en pharse de 3mots*

Experimentations et Résultats

Pour l'evaluation ils ont utilisé ASTD qui contient 10006 tweets mais ils n'ont selectioné que les tweet avec les classes "positive , negative et neutral" avec une totalité de 3315 tweets.
Les meilleurs resultats sont obtenus par RNTN avec lemmatisation ; 58,5% Accuracy et 53,6% F1-score.

Pour ce travail ,Heikal et al. [2018] ont tout d'abord fait passer les tweets par une phase de prétraitement et de nettoyage pour supprimer les symboles

et jetons indésirables, puis les tweets sont prêts pour la phase de formation. En suite , après avoir entraîné les deux modèles (CNN et LSTM)avec différents hyper-paramètres, ils ont sélectionné les deux meilleurs modèles qui donnent le F1-score le plus élevé tout en construisant un modèle d'ensemble.Ce modèle est utilisé pour prédire le sentiment final des tweets entrants.

Prétraitement des données

Dans cette phase,ils ont nettoyé les tweets tout en supprimant le bruit et les symboles indésirables. Cela est fait en supprimant les dates ,l'heure,les chiffres les URL ,les signes diacritiques arabes,l'élongation et utilisez plutôt une seule occurrence ; et aussi en normalisant les mots et supprimant les signes de ponctuation et les symboles indésirables .

Puisque chaque tweet a une longueur variable donc il a été représenté par un vecteur 2D de dimension $n * d$.

Ils ont utilisé le modèle AraVec skip gram , pour que les tweets soient de la même taille fixe,en complétant chaque représentation de tweet par des zéros. Donc chaque tweet sera de taille $n * d'$

CNN Modèle,Expérimentations,Résultats

Modèle

Le modèle se compose de trois CNN parallèles sous-modèles. Chaque sous-modèle a une certaine taille de filtre s et un certain nombre de filtres, m.
Ils ont appliqué ensuite le maximum global regroupement sur les cartes de caractéristiques obtenues. Il en résulte un vecteur de taille m (valeur maximale sélectionnée à partir de l'entité carte de chaque filtre).
La sortie des 3 sous-modèles est concaténée pour donner un vecteur de taille 3 m. Ils est suivi d'une couche entièrement connectée avec la fonction d'activation ReLU ;Une couche d'abandon. Enfin, une couche softmax avec trois unités de sortie est utilisée pour prédire le sentiment du tweet.

Expérimentation

Dans chaque expérience,ils ont changé l'une des valeurs d'hyper-paramètre tout en gardant les autres configurations fixes comme valeur par défaut.
Les hyper-paramètres du CNN par défaut sont répertoriés dans la figure suivante.

Hyper-parameter	Value
Filter sizes	[3, 4, 5]
Number of filters	200
Number of units in fully connected layer	30
Dropout rate	0.5
Learning rate	0.001
Number of epochs	10
Batch size	50

TABLE 3.1 – *Paramètre de configuration du CNN par défaut Heikal et al. [2018]*

Résultats

Comme meilleur résultat ils ont trouvé que :$Accuracy = 64,3\%$ et $F1 - score = 64,09\%$ avec un modèle complètement connecté de taille 100.

LSTM Modèle,Expérimentations,Résultats

Model

Un LSTM bidirectionnel est utilisé. Les tweets sont d'abord transmis à chacun des LSTM, chaque LSTM est de taille cachée h. La sortie finale de chaque LSTM est concaténé pour donner un vecteur de longueur 2h. Qui est ensuite transmis à une couche entièrement connectée avec ReLU. Une couche de suppression est placée après la couche LSTM et une autre est placée après la couche connectée.
Enfin, une couche softmax est ajoutée pour donner la classe de sentiment du tweet.

Expérimentation

Ils ont varié les paramètres cité dans la figure suivante séparément et ils ont calculé l'accuracy et F1-score.

Hyper-parameter	Value
LSTM hidden state dimension	200
Number of units in fully connected layer	30
Dropout rate	0.5
Learning rate	0.001
Number of epochs	10
Batch size	50

TABLE 3.2 – *Paramètre de configuration du LSTM par défaut Heikal et al. [2018]*

Résultats

Comme meilleur résultat ils ont trouvé qu'en utilisant le Dropout rate=0.2 on trouve que l'accuracy est à 64.75% et F1-score est à 62,08%

Modèle ensembliste et évaluation

Ils ont défini un modèle d'ensemble à partir du meilleur modèle CNN et du meilleur modèle LSTM. et les résultats sont regroupés dans la figure qui suit .

Model	Accuracy (%)	F1-score (%)
CNN (fully connected layer size=100)	64.30	64.09
LSTM (dropout rate=0.2)	64.75	62.08
Ensemble model	**65.05**	**64.46**

TABLE 3.3 – *Comparaison des résultats des différents modèles Heikal et al. [2018]*

Un amalgame des Benchmarks sité dans la table 3.4 était l'idée de Omara et al. [2018] en lui appliquant certains pretraitements tels que :la suppression des URL ,les lettres repétées et elongation de mots ,stop word,en normalisant certaines lettres, ensuite faire une tokenisation .
Au final le dataset contient (307425) de tokens .

Name	Total Entries	Source	PN* Entries	Description
TDS	2000	Tweeter	2000	1000 P 1000 N
ASTD	10006	Tweeter	2483	799 P 1684 N
SemEval	671	Tweeter	350	222 P 128 N
Social Media Posts	3200	Tweeter / Blogs	2479	719 P 1760 N
OCA	500	Movie Reviews	500	250 P 250 N
LABAR	63257	Book Reviews	51056	42832 P 8224 N
LARGE	34492	Reviews	31598	24948 P 6650 N
Health Services	2026	Tweeter	2026	628 P 1398 N

TABLE 3.4 – *Datasets pour l'analyse des sentiments arabes Omara et al. [2018]*

Ils ont fait une comparaison des algorithmes de ML ; KNN,SVM,Random Forest et autres avec l'architecture CNN du DL tout en variant les paramètres ,comme la table 3.5 le montre .

Layer	Output	Input Channels	Kernel	Features	Parameters
conv1d_1	1,008	37	7	256	66,560
max_pooling1d_1	336				
conv1d_2	330	256	7	256	459,008
max_pooling1d_2	110				
conv1d_3	108	256	3	256	196,864
conv1d_4	106	256	3	256	196,864
conv1d_5	104	256	3	256	196,864
conv1d_6	102	256	3	256	196,864
max_pooling1d_3	34				
flatten_1	8704				
dense_1	1,024	Weights + 1024 Bias			8,913,920
activation_1	1024				
dropout_1	1024				
activation_2	1024				
dense_2	1,024	Weights + 1024 Bias			1,049,600
activation_3	1024				
dropout_2	1024				
activation_4	1024				
dense_3	1	Weights + 1 Bias			1,025
activation_5	1				
Total Parameters					**11,277,569**

TABLE 3.5 – *Dimensions , paramètres et output du réseau (256 FEATURE MAPS) Omara et al. [2018]*

Resultats L'amélioration des performances en utilisant CNN est bien claire par rapport aux résultats trouvés en appliquant les algorithmes de Machine learning .
Les deux tables qui suivent résument les résultats trouvés.

Classifier	Processed Dataset			Raw Dataset		
	TF	Unigram	Bigram	TF	Unigram	Bigram
Nearest Neighbors	78.31	73.73	74.95	78.30	65.35	74.58
Support Vector Machine	86.21	77.79	77.23	85.85	77.11	76.88
Decision Tree	81.12	81.25	79.84	80.20	79.57	79.07
Random Forest	85.82	85.38	83.84	83.77	83.87	83.05
Multinomial Naive Bayes	87.09	78.95	78.42	86.15	78.29	77.75
Logistic Regression	86.79	81.26	77.26	85.58	80.21	76.90
Bernoulli Naive Bayes	81.95	81.95	77.24	81.16	81.16	76.99

TABLE 3.6 – *Resultats de l'approche ML Omara et al. [2018]*

Deep CNN	Dataset	Accuracy
[12] (256 Feature Maps)	**Processed**	92.90
	Raw	92.50
[12] (1024 Feature Maps)	**Processed**	94.33
	Raw	94.12
[11] 2*(64, 128, 256, 512) Feature Maps	**Processed**	94.00
	Raw	93.88
[11] (64, 128, 256, 512, 1024) Feature Maps	**Processed**	94.18
	Raw	93.85

TABLE 3.7 – *Résultats de l'approche DL(CNNs) Omara et al. [2018]*

Le travail de Lulu et Elnagar [2018] consiste à détecter automatiquement le dialecte utilisé qu'il soit Egyptien, Lenventin ou dialecte du Golf , cela en

utilisant un dataset AOC(Arabic Online Commentary) qui contient environ 110K de données, ils ont choisi d'utiliser que 33K(30K pour l'apprentissage et 3K pour l'évaluation).
Ils ont sélectionné 3 classes (parmi toutes les classes qui existent) qui sont "EGP,LEV,GLF". Ils sont partis du principe qu'une région utilise le même dialecte, d'où vient l'idée de fusionnement du dialecte Iraq et du Golf car ils appartiennent à la même région géographique et ils partagent presque le même dialecte .

Pré-traitement

Pour cette phase ils ont :

1. Enlevé tous les caractères non arabe
2. Éliminé les diacritiques et l'élongation .
3. Supprimé la ponctuation et les espaces

Ils n'ont pas implémenté la normalisation car ils supposent que cela peut affecter le sens du contexte.

Modèle et expérimentation

— Long-Short-Term-Memory (LSTM)
— Convolutional Neural Networks (CNN)
— Bidirectional LSTM (BiLSTM)
— Convolutional LSTM (CLSTM)

Ils sont utilisés comme classificateurs ,3 premières expérimentations ont été faites en prenant des paires de dialectes (classifications binaire).La 4^{ieme} expérimentation a été faite sur les 3 dialectes(3 classes).
Pour l'évaluation ils ont pris 80% pour l'apprentissage et 10% pour Cross-Validation et 10 % pour le test, la figure suivante résume les résultats d'Accuracy obtenus,où on remarque que BiLSTM apporte le meilleur résultat avec la paire EGP-GLF.

DA pair model	LSTM		CNN		BLSTM		CLSTM	
	CV	Test	CV	Test	CV	Test	CV	Test
EGP-GLF	97.1%	80.4%	96.8%	81.0%	93.3%	**83.8%**	89.6%	79.3%
EGP-LEV	91.3%	**81.3%**	96.0%	78.1%	87.7%	79.6%	90.8%	79.0%
LEV-GLF	91.7%	**77.2%**	96.8%	75.1%	95.3%	74.5%	89.5%	73.0%

TABLE 3.8 – *Résultat d'Accuracy pour 3 pairs de dialectes*
Lulu et Elnagar [2018]

Évaluation et discussion

La figure qui suit résume les précisions des classificateurs ternaires sur les 3 dialectes. Comme ils ont prévu, la précision est inférieure à ce qu'ils ont obtenu dans le tableau précédent. LSTM a atteint la plus grande précision

avec Accuracy = 71,4% suivi de CLSTM avec 71,1% et BLSTM avec 70,9%. Ils ont constaté que le modèle CNN souffre d'un problème de overfitting,la figure suivante nous montre clairement la différence entre les résultats de Cross Validation et ceux du test.

Model	CV	Test
LSTM	84.5%	**71.4%**
CNN	96.0%	68.0%
BLSTM	84.4%	70.9%
CLSTM	90.4%	71.1%

TABLE 3.9 – *Résultat d'Accuracy pour 3 dialectes ensemble Lulu et Elnagar [2018]*

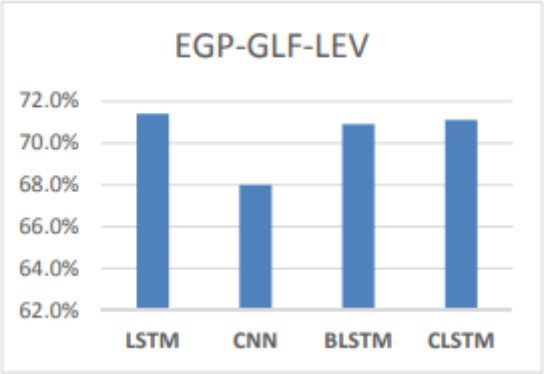


FIGURE 3.4 – *Accuracy pour 3 dialectes ensemble avec les différents classificateurs.*

Mohammed et Kora [2019] s'intéressent à l'analyse des sentiments des twitteurs , ils ont collecté leur propre dataset en utilisant Twitter API et ils ont classifié les tweets après la collecte,la figure qui suit résume les informations sur le corpus .

Total number of tweets	40,000
Number of positive tweets	20,000
Number of negative tweets	20,000
Number of words	359,818
Max tweet token	39
Number of tokens	1,953,869
Average tokens per tweet	17

TABLE 3.10 – *Statistique du Corpus Mohammed et Kora [2019]*

Prétraitement :

- La suppression des re-tweet était nécessaire , ainsi que les tweets de spam qui continent des publications ou des liens inappropriés .
- Les tweets qui contiennent d'autres dialectes tels que Khaliji était carrément enlevés .
- Les diactrics ,lettres ou mot non-arabe,l'élongation de mots sont supprimés.
- Une standardisation et normalisation des lettres .
- En dernier , ils ont corrigé manuellement les mots qui n'étaient pas bien écrit(lors d'un oubli d'une lettre par exemple).

Pour la représentation des mots ils utilisent Word2Vec avec le modèle CBOW qui est déjà entraîné

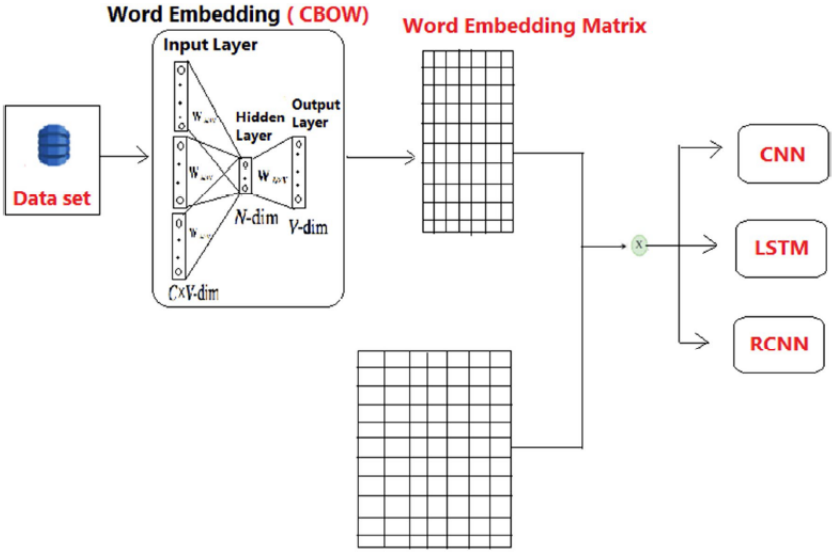


FIGURE 3.5 – *Représentation de mot Word embedding*
Mohammed et Kora [2019]

Architecture utilisée :

CNN

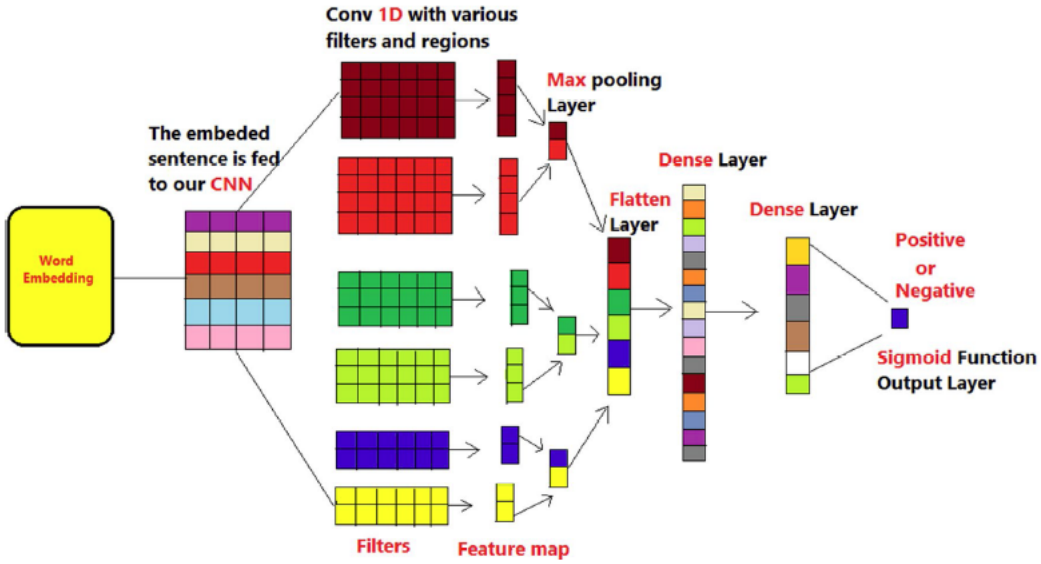


FIGURE 3.6 – *Architecture de CNN Mohammed et Kora [2019]*

LSTM

Après avoir appliquer la couche d'incorporation de mots, les mots incorporés sont introduits dans les cellules LSTM. Les cellules LSTM seront entraînées sur les mots incorporés et produiront leurs mots de prédiction, qui sont entièrement connectés avec une couche dense de type sigmoïde .

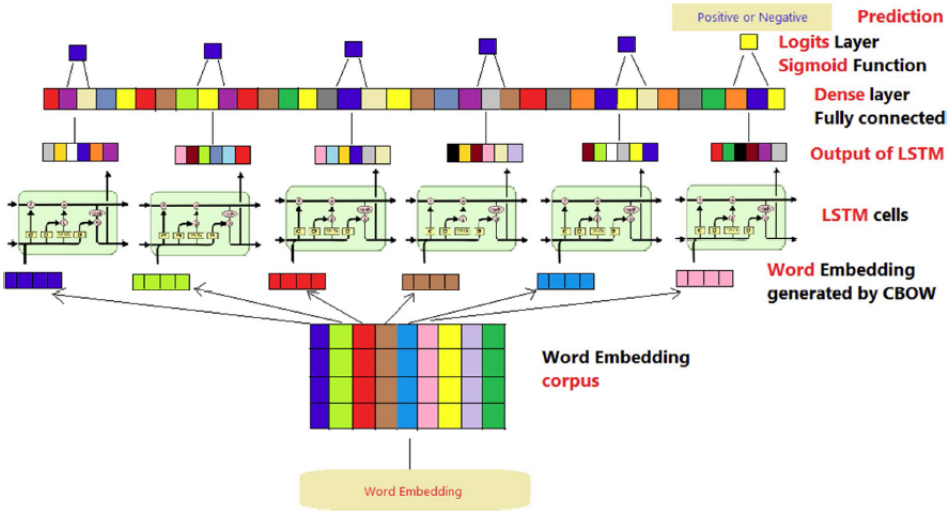


FIGURE 3.7 – *Architecture de LSTM Mohammed et Kora [2019]*

RCNN

RCNN est composé d'une couche de convolution, max pooling, flatten, LSTM et dense layers.

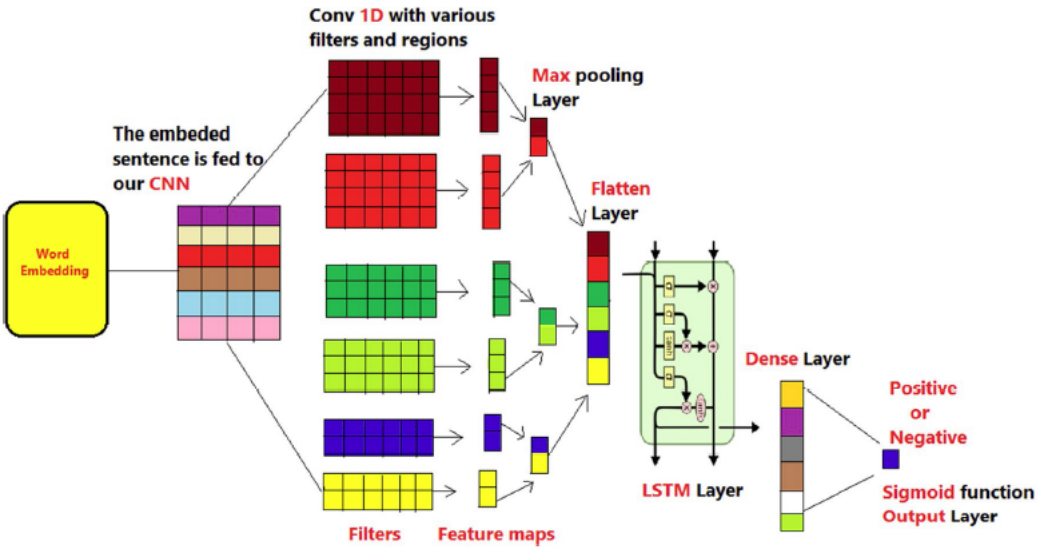


FIGURE 3.8 – *Architecture de RCNN Mohammed et Kora [2019]*

Résultats :

Models	AVG accuracy (%)	AVG recall (%)	AVG precision (%)	AVG f-score (%)
CNN	75.72	74.60	78.03	75.06
LSTM	81.31	80.92	81.99	81.25
LSTM+Aug	**88.05**	**88.9**	**86.94**	**87.24**
RCNN	78.46	77.38	80.27	77.86

TABLE 3.11 – *Résumé des résultats Mohammed et Kora [2019]*

LSTM avec augmentation des données ,atteint la plus grande accuracy et le f1-score le plus élevé avec des valeurs de 88,05% et 87,24%, respectivement. Il convient de mentionner que l'application des données augmentée sur les deux modèles CNN et RCNN n'a pas d'effet significatif. Cela est dû au fait d'adopter l'idée du shuffling aléatoire comme technique d'augmentation, où l'ordre des mots est modifié dans le corpus. Cet ordre est fortement pris en compte avec LSTM puisqu'il s'agit de RNN et dépend également de la position de chaque mot dans les phrases.
CNN et RCNN, ont besoin de plus de données réelles autres que le corpus arabe pour améliorer les résultats.

Une hybridation de CNN et LSTM a été proposé par Farha et Magdy [2019].
Tout d'abord une étapes de prétraitement est prérequise :

— En normalisant certaines lettres.
— En éliminant les élongations de mots .
— En supprimant les caractères non arabes,diactrics,ponctuation et URL.

Pour la représentation de mot ils ont choisi une représentation à 2 dimensions (matrice) qui correspond au word embedding (utilisation de AraVec).
L'architecture proposé prend en entrée word embeddings qui sont liés à une couche de convolution 1D suivit par Max Pooling Layer ,cette dernière est liée au LSTM suivit par dense Layer et une application de Softmax pour avoir le résultat final en output Layer .
Et cela s'effectue en utilisant les hyper-paramètres qui se résument sur la table suivante

Parameter	Value
#LSTM cells	128
Recurrent dropout	20%
Output dropout	20%
#Filters	300
Filter size	3
Pooling size	2
Optimizer	Adam
Learning rate	0.0001
Activation	ReLU

TABLE 3.12 – *CNN-LSTM model hyper-parameters Farha et Magdy [2019]*

Pour valider et prouver leur approche ils l'ont appliqué sur 3 dataset :

1. SemEval (Benchmark dataset)contient 6100 instances pour l'apprentissage et 3555 pour le test avec les classes "Positive,Negative,Neutral"
2. ASTD (Benchmark dataset) contient 10006 tweets, 6691 parmis eux étaientt supprimés car ils sont de classe "objective" et qui n'aide pas dans l'analyse des sentiments.
3. ArSAS contient 21K tweets avec les classes "Positive,Negative,Neutral,Mixed" les instances de cette derniére classe ont été supprimé car elles étaient minimes.

La table 3.13 résume les résultats trouvés ,on remarque que leur modèle arrive à améliorer les résultats trouvé au paravent par Heikal et al. [2018] ;et surtout en l'appliquant ArSAS qui contient des données massives l'accuracy est à 0.92.

Dataset	*System*	*AvgRec*	F^{PN}	*Acc*
SemEval	(El-Beltagy et al., 2017)	0.58	0.61	0.58
	Mazajak	**0.61**	**0.63**	**0.62**
ASTD	(Heikal et al., 2018)	0.61	0.71	0.65
	Mazajak	**0.62**	**0.72**	**0.66**
ArSAS	Mazajak	**0.90**	**0.90**	**0.92**

TABLE 3.13 – *Résumé des résultats trouvé Farha et Magdy [2019]*

3.2 Détection de Hate Speech arabe

Albadi et al. [2018] ont créé leur propre dataset en utilisant l'API de twitter pour la détection de Hate Speech de religion.
L'idée du prétraitement consiste à faire la normalisation comme les autres ont fait précédemment(Enlever les Hashtags ;underscore ;diacritics et l'élongation de mot) mais ils ont pris en considération que la suppression des mots négatifs parmi les mots vides peuvent influencer sur le sens du tweet , ce qui n'a pas été pris en considération dans les autres travaux .
MADAMIRA a été utilisé comme lemmatizer par contre Alshalan et Al-Khalifa [2020] ont utilisé un autre qui est plus optimale .

Experimentation

GRU-based network avec pre-trained word embedding a été utilisé ;tout en passant par l'indexation des données en mots uniques (padding par 0, la longeur maximale pour représenter le mot est 50) .
Après ils passent au embedding layer (Twitter CBOW 300-dimensions) ;la sortie de cette couche est un embedding vecteur de taille "50,300" passé au dropout layer .
GRU layer avec 240 hidden units au finale une couche de sortie (en utilisant sigmoïd) .

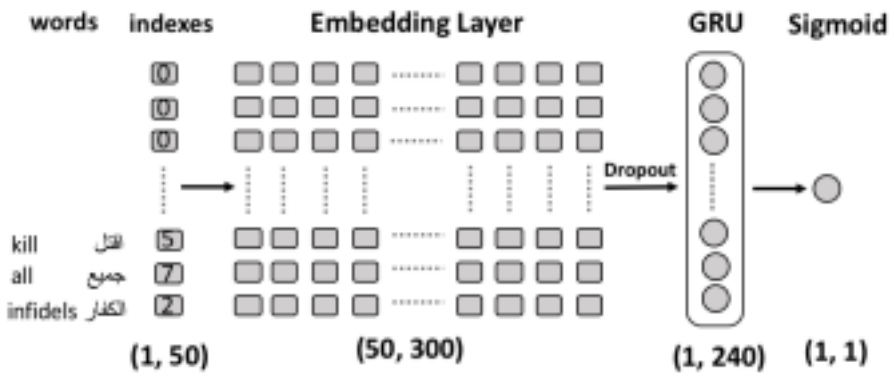


FIGURE 3.9 – *L'architecture GRU utilisée par Albadi et al. [2018]*

Résultats

GRU-based RNN avec pretrained word embedding a donné les meilleures performances avec 0.79 d'accuracy et 0.84 AUROC. Chowdhury et al. [2019] se sont basés sur l'idée derrière l'exploitation de l'interaction avec la communauté est que si nous avons des informations sur les membres d'une communauté définie par une mesure de similitude, on peut alors déduire des informations sur une personne en fonction de quelle communauté ils appartiennent.
Pour le prétraitement ils ont utilisé les mêmes étapes que ceux de Albadi et al. [2018] sauf qu'ils ont utilisé ISRI arabic stemmer proposé par NLTK.
Le meilleur résultat d'accuracy était 0.81 obtenu par Bi-GRU+CNN+NODE2VEC(avec 240 units ;2 convolutionel layers) et qui a prouvé son efficacité et il a amélioré les résultats de l'étude précédente (Accuracy a 0.79 pour Albadi et al. [2018])

Mohaouchane et al. [2019] ils s'intéressent à la détection du langage offensif arabe sur la platforme ou réseau social Youtube.Ils ont utilisé le dataset de Alakrot et al. [2018] qui a fait a son tour l'objet de sa thèse Alakrot [2019].
Pour le prétraitement ils n'ont gardé que les caractères arabe , ils ont supprimé tout ce qui est caractères spéciaux ,diactrics ,ponctuation et élongation.
Aprés avoir effectuer une tokenisation ils choisissent de représenter les mots en utilisant AraVec ,SkipGram (ils suppose que le SK donne de meilleur résultats par rapport a CBOW).
Ils ont utilisé différents modèles (CNN,BiLSTM,Attention BiLSTM,CNN-LSTM) comme il est illustré dans la figure 3.10 avec différents paramètres pour chaque modèle .

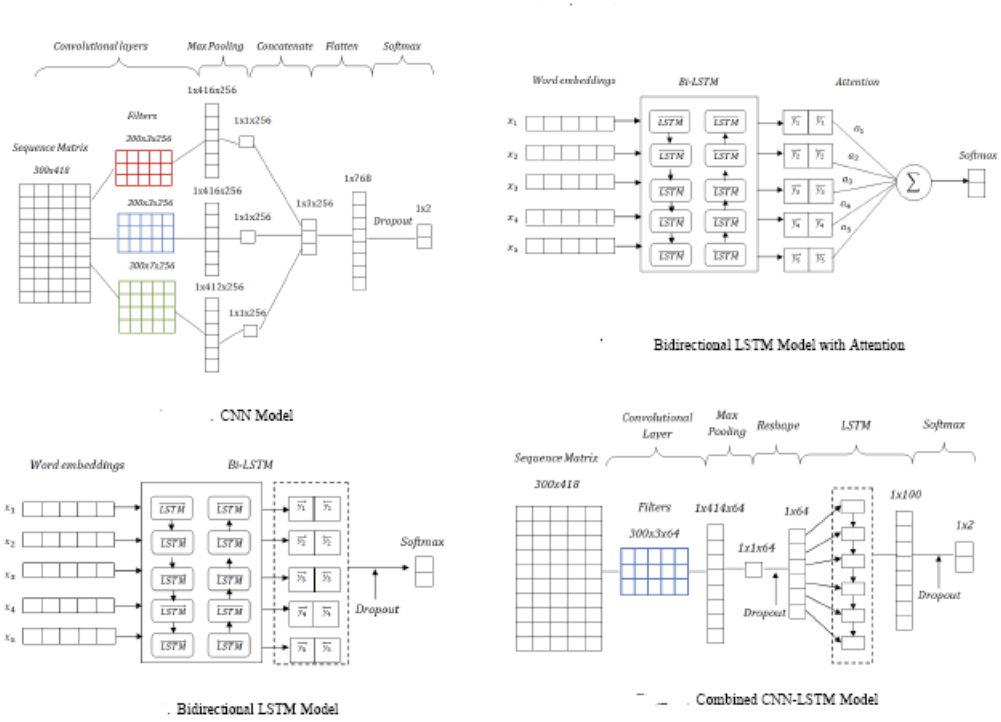


FIGURE 3.10 – *Différents modèles proposés par Mohaouchane et al. [2019]*

Après qu'ils ont essayé les différents paramètres ils ont constaté que l'hybridation de CNN-LSTM a plus de pouvoir pour rappeler les commentaires qui contiennent du hate ou pas (rappel=83,46).
D'une autre part le CNN a une précision =86,1 et F1-score =84,05 et accuracy=87,84.

	Accuracy	Precision	Recall	F1-Score
CNN	**87.84**	**86.10**	82.24	**84.05**
Bi-LSTM	86.42	83.74	80.97	82.33
Attention Bi-LSTM	85.75	82.07	81.51	81.70
Combined CNN-LSTM	87.27	83.89	**83.46**	83.65

TABLE 3.14 – *Résultats des différents modèles proposés par Mohaouchane et al. [2019]*

Faris et al. [2020] ont collecté leur propre dataset en utilisant Twitter API ; ce dataset inclue plusieurs débats(Religion , racisme ...).
Le nombre total des tweets est 3696(Hate :843,Normal :791 ;Neutral 2062), sauf les instances de Hate et Normal étaient traitées pour cette étude.

La phase de prétraitement , la suppression des caractères non arabe , les nombres ,les symboles ,la ponctuation , hashtags,diactrics était nécessaires à leurs avis.
Aussi une normalisation était faite pour certaine lettre ,la tokenisation est

faite a l'aide de NLTK.
Les mots ont été représenté par le Word2Vec , et aussi en utilisant AraVec qui est fournit par NLTK pour les comparer ensuite .

Architecture utilisée :
- Commence par Embedding Layer ,
- Dropout Layer pour éviter l'overfitting ,
- Convolutional Layer ,
- Max Pooling Layer,
- Ensuite LSTM Layer,
- Fully connected Layer,
- Au final Output Layer(sigmoid).

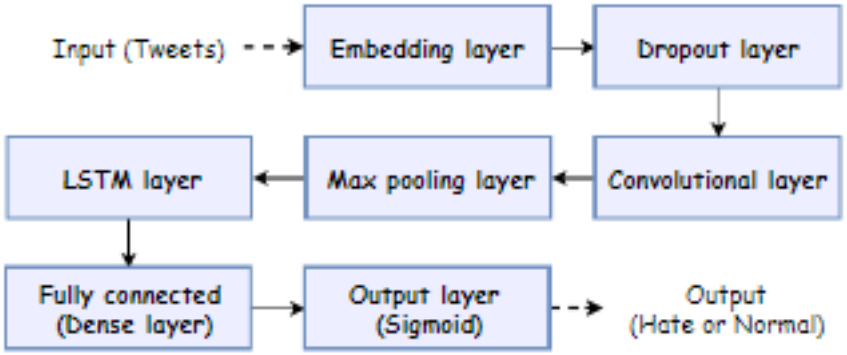


FIGURE 3.11 – *L'architecture proposée Faris et al. [2020]*

Résultat :
En utilisant AraVec(N-grams et SG) avec embedding de dimension 100 et en appliquant leur architecture (hybridation CNN et LSTM) ;ils aboutissent a 66.54% d'accuracy et 79,76 % Recall ; 71.68 pour F1-score.

Le Hirak désigne une série de manifestations sporadiques qui ont lieu depuis le 16 février 2019 en Algérie pour protester dans un premier temps contre la candidature d'Abdelaziz Bouteflika à un cinquième mandat présidentiel, tous les sujets tournaient autour du Hirak pendant cette période sur les réseaux sociaux , y avait un tas d'informations qui ont servi à Guellil et al. [2020] pour faire leurs études sur cet évènement .
Les commentaires des vidéos sur YouTube qui concernent le Hirak étaient la source d'information pour construire leur dataset,en utilisant Youtube API, 3 annotations étaient appliquées pour pouvoir décider s'il s'agit du Hate (1) ou non-Hate (0).

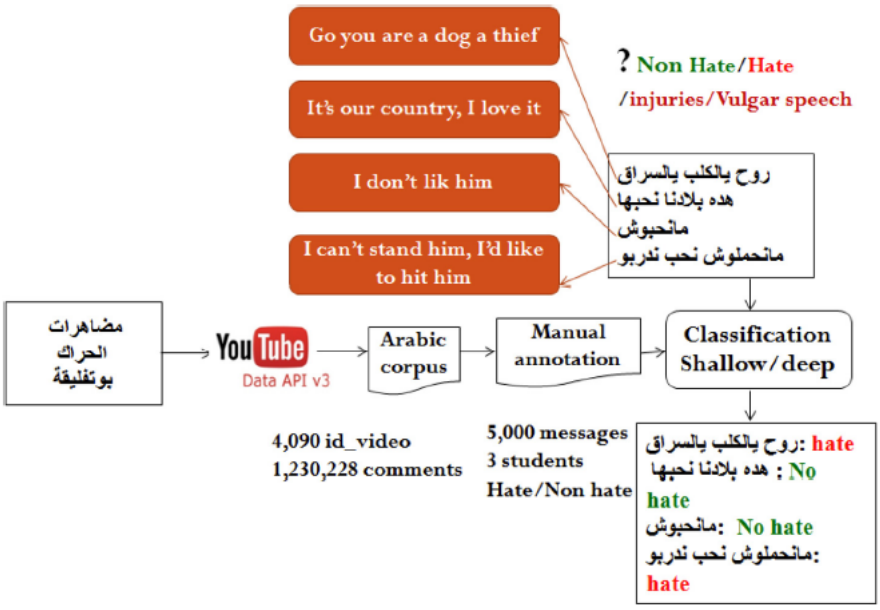


FIGURE 3.12 – *Détection de hate speech dans un débat politique arabe Guellil et al. [2020].*

Pour améliorer la précision,ils se sont concentrés sur la partie du corpus où tous les annotateurs avait la même annotation. Le corpus construit ,contient 3384 commentaires (où 2465 sont classés non-Hate et 919 sont classés étant Hate). Ils ont également construit un corpus équilibré contenant 2000 des commentaires choisis au hasard dans le corpus mentionné .
Ils proposent d'utiliser Word2Vec et fastText pour mieux représenter les mots .
La figure suivante nous montre les paramètres appliqués aux architectures du Deep Learning.

Model	Layers	Output shape	Params
CNN	embedding_1(SG/CBOW)	(None,37,300)	5106900
	conv1d_d	(None,37,300)	630300
	global_max_pooling1d_1	(None,300)	0
	dropout_1	(None,300)	0
	dense_1	(None,600)	180600
	dense_2	(None,2)	1202
MLP	embedding_1(SG/CBOW)	(None,37,300)	5106900
	dense_1	(None,37,64)	19264
	global_max_pooling1d_1	(None,64)	0
	dropout_1	(None,64)	0
	dense_3	(None,600)	39000
	dense_4	(None,2)	1202
LSTM	embedding_1(SG/CBOW)	(None,37,300)	510690
	lstm_1	(None,37,64)	93440
	global_max_pooling1d_1	(None,64)	0
	dropout_1	(None,64)	0
	dense_1	(None,600)	39000
	dense_2	(None,2)	1202
Bi-LSTM	embedding_1(SG/CBOW)	(None,37,300)	5106900
	bidirectional_1	(None,37,128)	186880
	global_max_pooling1d_1	(None,128)	0
	dropout_1	(None,128)	0
	dense_1	(None,600)	77400
	dense_2	(None,2)	1202

TABLE 3.15 – *Architecture des modèles du Deep Learning*
Guellil et al. [2020]

Résultats

Models	Type	ML Alg	Hateful			Non-hateful			Average		
			P	R	F1	P	R	F1	P	R	F1
Word2vec	SG	GNB	0.47	0.71	0.56	0.85	0.68	0.75	0.74	0.68	0.70
		LR	0.94	0.46	0.62	0.82	0.99	0.90	0.85	0.84	0.82
		RF	0.83	0.46	0.59	0.82	0.96	0.88	0.82	0.82	0.80
		SGD	0.91	0.63	0.75	0.87	0.98	0.92	0.88	0.88	0.87
		LSVC	0.89	0.78	0.83	0.92	0.96	0.94	0.91	0.91	*0.91*
	CBOW	GNB	0.48	0.71	0.57	0.85	0.69	0.76	0.75	0.69	0.71
		LR	0.86	0.51	0.64	0.83	0.97	0.89	0.84	0.84	0.82
		RF	0.82	0.48	0.61	0.82	0.96	0.89	0.82	0.82	0.81
		SGD	0.87	0.64	0.74	0.87	0.96	0.91	0.87	0.87	0.86
		LSVC	0.84	0.71	0.77	0.89	0.95	*0.95*	0.87	0.88	0.87
fastText	SG	CNN	0.89	0.75	0.82	0.91	0.96	0.93	0.90	0.90	0.90
		MLP	0.88	0.81	*0.84*	0.93	0.95	0.94	0.91	0.91	*0.91*
		LSTM	0.83	0.79	0.81	0.92	0.94	0.93	0.89	0.89	0.89
		Bi-LSTM	0.83	0.78	0.81	0.91	0.94	0.93	0.89	0.89	0.89
	CBOW	CNN	0.79	0.70	0.74	0.88	0.93	0.90	0.86	0.86	0.86
		MLP	0.65	0.85	0.73	0.93	0.81	0.87	0.85	0.82	0.83
		LSTM	0.82	0.68	0.74	0.88	0.94	0.91	0.86	0.87	0.86
		Bi-LSTM	0.79	0.74	0.76	0.90	0.92	0.91	0.87	0.87	0.87

Models	Type	ML Alg	Hateful			Non-hateful			Average		
			P	R	F1	P	R	F1	P	R	F1
Word2vec	SG	GNB	0.67	0.70	0.69	0.71	0.68	0.69	0.69	0.69	0.69
		LR	0.89	0.79	0.84	0.82	0.91	0.86	0.85	0.85	0.85
		RF	0.85	0.65	0.74	0.73	0.89	0.80	0.79	0.78	0.77
		SGD	0.92	0.73	0.81	0.79	0.94	0.85	0.85	0.83	0.83
		LSVC	0.88	0.85	*0.87*	0.87	0.89	*0.88*	0.87	0.87	*0.87*
fastText	CBOW	GNB	0.71	0.70	0.71	0.73	0.73	0.73	0.72	0.72	0.72
		LR	0.86	0.75	0.80	0.79	0.88	0.84	0.82	0.82	0.82
		RF	0.83	0.68	0.75	0.74	0.87	0.80	0.79	0.78	0.78
		SGD	0.96	0.62	0.75	0.73	0.98	0.84	0.84	0.81	0.80
		LSVC	0.85	0.82	0.84	0.84	0.86	0.85	0.85	0.84	0.84
	SG	CNN	0.89	0.79	0.84	0.82	0.91	0.86	0.86	0.85	0.85
		MLP	0.87	0.82	0.84	0.84	0.88	0.86	0.85	0.85	0.85
		LSTM	0.93	0.69	0.80	0.77	0.95	0.85	0.85	0.83	0.82
		Bi-LSTM	0.93	0.79	0.85	0.83	0.94	*0.88*	0.87	0.87	*0.87*
	CBOW	CNN	0.86	0.78	0.82	0.81	0.88	0.84	0.83	0.83	0.83
		MLP	0.70	0.88	0.78	0.85	0.65	0.73	0.78	0.76	0.75
		LSTM	0.86	0.76	0.81	0.80	0.88	0.84	0.82	0.82	0.82
		Bi-LSTM	0.86	0.78	0.82	0.81	0.88	0.85	0.83	0.83	0.83

TABLE 3.16 – *Résultats après application sur le corpus 1 et 2(respectivement) Guellil et al. [2020].*

Les résultats obtenus nous montrent qu'en utilisant Skip Gram(SG) modèle au BiLSTM,MLP on a l'accuracy à 91%.

Les principales contributions du travail de Alshalan et Al-Khalifa [2020] sont triples :

1. Construire un ensemble de données publiques de 9316 tweets étiquetés comme haineux, abusifs et normaux.
2. Comparaison des performances de trois modèles de réseaux neuronaux - CNN, GRU, CNN + GRU - pour la tâche de détection des discours haineux en arabe.

3. Évaluation du modèle récent de représentation linguistique BERT pour le discours de haine arabe tâche de détection.

Dataset utilisé

L'ensemble des données contient un total de 9316 tweets classés comme hateful, abusive, our normal (non haineux ou abusive). Leur tâche était d'effectuer une classification binaire pour classer le tweet en discours haineux ou non haineux, ils ont gardé que les tweets annotés comme haineux ou normaux.
Donc ils ont obtenu 8964 tweets répartis entre 75 et 25% pour la formation (6425 instances) et les tests. (2539 instances). Ils ont utilisé les 75% pour la formation et le réglage des hyper-paramètres par cross validation puis ils ont testé les modèles optimisés sur les 25% held-out data.

Prétraitement

— Suppression des hashtags
— Filtrage du spam
— balises que nous avons trouvées
— Description des emojis
— Nettoyage :la ponctuation, les espaces blancs supplémentaires, les signes diacritiques et les caractères non arabes sont supprimés.
— Lemmatisation : en utilisant le stemmer de Farasa.

Architectures utilisées et résultats

CNN

Contient cinq couches :
Une couche d'entrée (couche d'intégration) ;
Une couche de convolution, qui consiste essentiellement en 3 couches de convolution parallèles avec différentes tailles de noyau (2, 3 et 4) ;
Une couche de mise en commun ;
Une couche dense cachée ;
Enfin la Couche de sortie.
Tous les tweets sont mappés dans des vecteurs à valeur réelle de 300 dimensions par la couche d'intégration.

GRU

Contient 4 couches :
Une couche d'entrée (couche d'intégration),
Une couche GRU,
Une couche cachée couche dense, et enfin
La couche de sortie.
Semblable à l'architecture CNN, les entrées (tweets) sont introduites dans le calque d'intégration, qui mappe les tokens des tweets dans un vecteur de valeur réelle de 300 dimensions.

CNN+GRU

La troisième architecture expérimentée est une combinaison de CNN et GRU. Dans cette architecture, CNN est utilisé comme un extracteur de fonctionnalités(feature extractor) qui transmet la «fonctionnalité extraite» à une Couche GRU, qui traite la dimension d'entité générée comme des pas temporels.
L'architecture CNN + GRU contient 6 couches : une couche d'entrée (couche d'intégration), une couche de convolution avec 100 filtres et un noyau taille de 4, une couche de max pooling , une couche GRU, une autre couche de pooling max, et enfin la couche de sortie. La couche d'intégration mappe les tweets dans un espace vectoriel de 300 dimensions.

Résultat

Model	GHSD (In-Domain)						RHSD (Out-Domain)					
	P	R	F1	A	HCR	AUROC	P	R	F1	A	HCR	AUROC
SVM (char n-grams)	0.74	0.74	0.74	0.78	0.63	0.85	0.70	0.66	0.66	0.68	0.46	0.73
LR (char n-grams)	0.75	0.74	0.75	0.79	0.63	0.84	0.68	0.66	0.65	0.68	0.47	0.72
CNN	**0.81**	**0.78**	**0.79**	**0.83**	**0.67**	**0.89**	0.72	**0.69**	**0.69**	**0.70**	**0.56**	**0.79**
GRU	0.80	0.77	0.78	0.82	0.63	0.87	0.70	0.65	0.64	0.68	0.43	0.76
CNN + GRU	0.80	0.76	0.77	0.82	0.62	0.88	**0.74**	0.67	0.67	**0.70**	0.44	0.77
BERT	0.76	0.76	0.76	0.80	0.65	0.76	0.63	0.60	0.59	0.63	0.38	0.6

TABLE 3.17 – *Résultats de l'évaluation des modèles expérimentésAlshalan et Al-Khalifa [2020]*

Les meilleurs résultats ont étaient obtenus par CNN avec accuracy 0.83 et 0.78 de rappel et F1-score a 0.79.

3.3 Tableau comparatif

Année	Auteurs	Citations	Dataset	Approche (Architecture)	Résultats
2015	Al Sallab et al.	82	LDC ATB	RAE	Acc=74,3% F1-score= 73,5%
2017	Baly et al.	31	Leur propre Dataset (train) ASTD(test)	RNTN avec lemma-tisation	Acc=58,5% F1-score= 53,6%
2018	Heikal et al.	65	ASTD	CNN+ LSTM	Acc=65,05% F1-score= 64,46%
2018	Omara et al.	9	Amalgame de Benchmark	CNN	Acc=94%
2018	Lulu et Elnagar	29	AOC	LSTM	Acc=71,4%
2018	Albadi et al.	55	Leur propre Dataset	GRU-based RNN	Acc=0.79
2019	Mohammed et Kora	19	Leur propre Dataset	LSTM+ augm	Acc=88,05% F1-score= 87,24%
2019	Farha et Magdy	37	SemEval ASTD ArSAS	CNN+ LSTM	Acc=0,92 (sur ArSAS)
2019	Chowdhury et al.	7	Dataset de Albadi et al.	BiGRU+ CNN+ NOD2VEC	Acc=0,81
2019	Mohaouchane et al.	11	Dataset de Alakrot et al.	Hyb CNN+ LSTM CNN	Acc=87,84% F1-score= 84,05% Precision= 86,1% Rappel=83,46
2020	Faris et al.	4	Leur propre Dataset	CNN+ LSTM	Acc=66,54% F1-score= 71,68%
2020	Guellil et al.		Leur propre Dataset	BiLSTM MLP	Acc=91%
2020	Alshalan et Al-Khalifa	2	Leur propre Dataset	CNN	Acc=0,83 F1-score= 0,79

Table 3.18 – *Tableau comparatif .*

Tous ces travaux importants concernant l'analyse des sentiments et le Hate Speech en arabe ,nous ont permis d'avoir une idée plus précise sur les techniques de pré-traitement ,les hyper-paramètres optimals ainsi que les architectures et modèles utilisés pour mieux performer par la suite dans notre approche .

Méthodologie 4

4.1 Corpus utilisé

Tous les travaux et les datasets utilisés qui parlent sur le Hate Speech sont représentés dans un même site hatespeechdata.com d'où nous avons tiré les datasets suivants .

4.1.1 Premier dataset

Dans l'intérêt de la communauté de recherche, Mubarak et al. [2017] ont publié un ensemble de données de 32K commentaires supprimés de Aljazeera.net.
Aljazeera.net,est une chaîne d'actualité arabe populaire .Il modère tous les commentaires qui apparaissent sur leur site [1]. Le commentaire n'est pas accepté s'il s'agit d'une attaque personnelle, raciste, sexiste ou autrement offensant, incitant violence, non pertinent, publicité, etc.
Au départ,ils ont obtenu un corpus de 400 000 commentaires sur environ 10 000 articles couvrant de nombreux domaines tels que la politique, l'économie, la société et la science. À partir de ces commentaires, ils ont sélectionné 32K commentaires dont la longueur est comprise entre 3 et 200 caractères pour faciliter les annotations ultérieurement.

Informations générales

- Le lien : https://bit.ly/3i3cNBn
- Taille de dataset : 32000
- Pourcentage du contenu abusive : 81%
- Langue : Arabe
- Plateforme : AlJazeera(Facebook,Twitter,YouTube)

Le dataset contient 3 classes , ou nous avons remarqué que la classe (-2 :abusive) n'a que 533 instances donc il pourra nuire au performance du coup on l'avait considéré comme hate avec la classe -1.

1. www.aljazeera.com

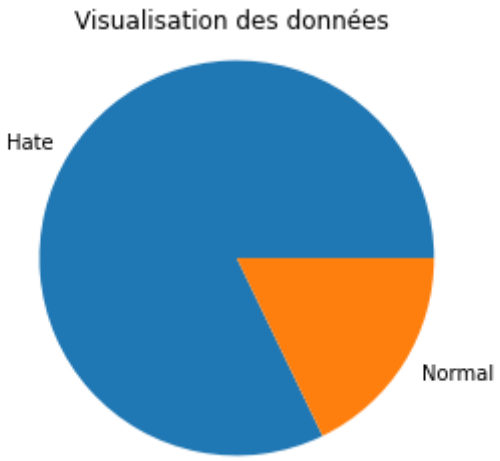


FIGURE 4.1 – *Visualisation du compartiment de classes*

4.1.2 Deuxième dataset

Il a été créé par Alakrot et al. [2018] . Ils ont utilisé un outil open source pour collecter les commentaires YouTube de Klostermann [2015] . La figure suivante représente l'interface de ce scraper.

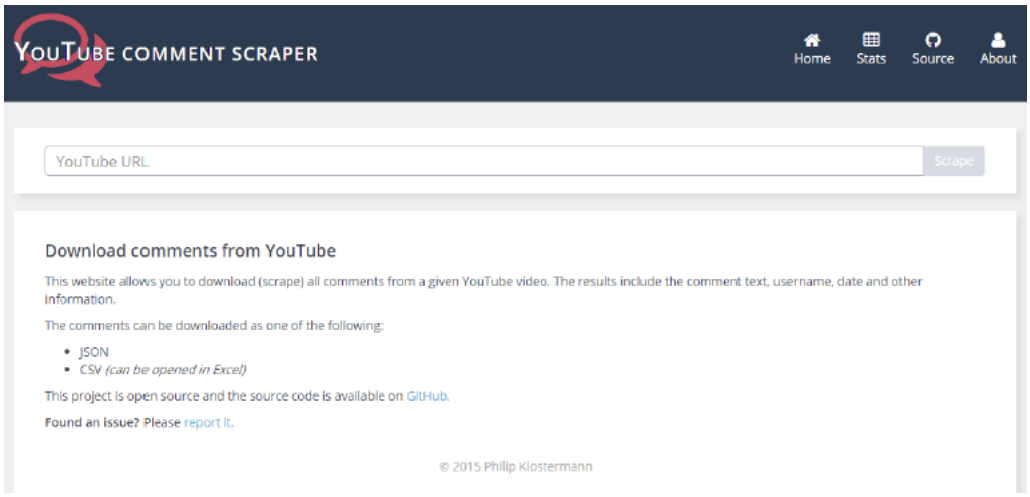


FIGURE 4.2 – *Interface de YouTube comment scraper*

Informations générales

- Le lien : `https://bit.ly/2TMF6Kx`
- Taille de dataset : 15050
- Pourcentage du contenu abusive : 39%
- Langue : Arabe
- Plateforme : YouTube

4.1.3 Troisiéme dataset

Ce dataset est un amalgame de deux dataset : L-HSAB dataset Mulki et al. [2019]. Il peut être décrit comme un ensemble de données politiques puisque la majorité des tweets ont été recueillie auprès du chronologies des politiciens, des militants sociaux / politiques et les ancres de télévision.

Informations générales sur le dataset de Mulki et al.

- Le lien : https://bit.ly/34xtCMM
- Taille de dataset : 5846
- Pourcentage du contenu abusive : 38%
- Langue : Arabe
- Plateforme :Twitter

Deuxiéme dataset construit par Mubarak et Darwish [2019] sur le réseau social Twitter .

Informations générales sur le dataset de Mubarak et Darwish

- Le lien : https://bit.ly/3z9qlBI
- Taille de dataset : 1100
- Pourcentage du contenu abusive : 59%
- Langue : Arabe
- Plateforme :Twitter

Puisque dans chaque dataset le nom de la classe est différent, on a procédé par une unification de classe de sort que tous les données(texte) soient classifiés en 1 (Hate) ou 0 (Normal).

4.2 Nettoyage de données

Regex : (contraction de regular expression) est une manière commune de nommer les expressions régulières. Ces puissants outils permettent de représenter des spécimens de chaînes de caractères c'est une librairie disponible dans python sous le nom **re**.

4.2.1 Suppression des diactrics

Les diactrics dans notre cas d'étude n'ont aucun apport pour la compréhension du sens . Nous nous intéressons seulement aux mots.

```
arabic_diacritics = re.compile("""
                                ّ    | # Tashdid
                                َ    | # Fatha
                                ً    | # Tanwin Fath
                                ُ    | # Damma
                                ٌ    | # Tanwin Damm
                                ِ    | # Kasra
                                ٍ    | # Tanwin Kasr
                                ْ    | # Sukun
                                ـ     # Tatwil/Kashida
                            """, re.VERBOSE)
```

FIGURE 4.3 – *Définition des diactrics sous Python*

4.2.2 Suppression des ponctuations

Les ponctuations tels que : " :","!",".","؟" et autre , n'ont aucun pouvoir expressif donc il vaut mieux les supprimer.

4.2.3 Suppression des balises HTML

Les balises ne donnent pas d'informations utiles pour notre cas d'étude.

4.2.4 Suppression des lignes vides

Puisqu'elles sont vides donc il est préférable de les supprimées

4.2.5 Suppression de tous les caractères non arabes

Comme nous nous intéressons au contenu arabe pas autre donc la suppression des caractères non arabe est une nécessité cela est fait a l'aide de la librairie Alphabet-Detector sous Python .

```
def keep_only_arabic(text):
    ad = AlphabetDetector()
    clean_lines = list()
    for line in text.splitlines():
        clean_line = list()
        for word in line.split():
            if len(word) > 1:
                if ad.is_arabic(word):
                    if word.isalpha():
                        clean_line.append(word)
        clean_lines.append(' '.join(clean_line))
    return '\n'.join(clean_lines)
```

FIGURE 4.4 – *Suppression des caractères non arabes sous Python*

4.2.6 Suppression des élongations

Le fait d'utiliser ou non une élongation de mot le sens ne change pas mais la taille diffère , puisque le sens du mot nous intéresse le plus ,leurs suppression est la meilleure solution.

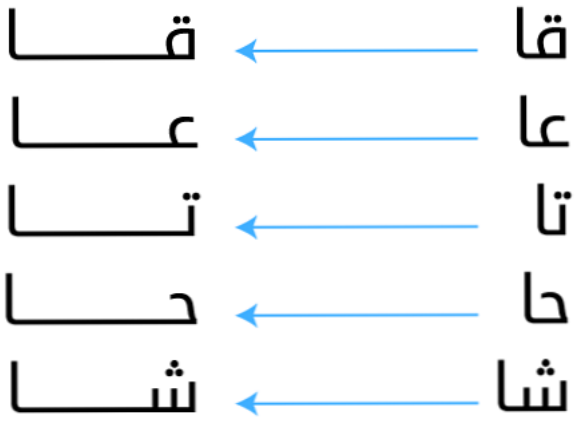


FIGURE 4.5 – *Exemple sur l'élongation*

4.3 Convolutional Neural Networks

Les réseaux de neurones convolutifs (**C**onvolutional **N**eural **N**etwork) sont à ce jour les modèles les plus performants pour **classer des images**. ils comportent deux parties bien distinctes.
En entrée, une image est fournie sous la forme d'une **matrice de pixels**. Elle a deux dimensions pour une image aux niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].
La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers d'une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées **cartes de convolutions** .Certains **filtres intermédiaires** réduisent la résolution de l'image par une opération de maximum local.
Enfin, les cartes de convolutions sont mises à plat et concaténées en **un vecteur** de caractéristiques, appelé **code CNN.**
Ce code CNN en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées. Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.
La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories.

Couche de convolution

La couche de convolution est le bloc de construction de base d'un CNN.Elle traite les données d'un champ récepteur .

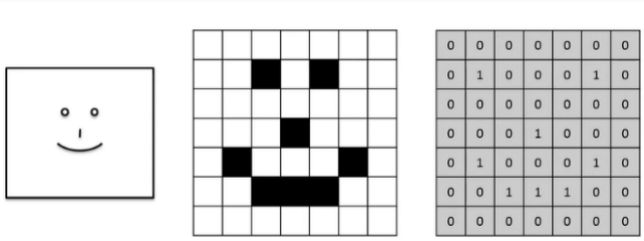


FIGURE 4.6 – *Exemple de convolution*

Max Pooling

Max Pooling aide à réduire la carte des caractéristiques afin de faire la classification plus précisément.Elle permet de compresser l'information en réduisant la taille de l'image intermédiaire .

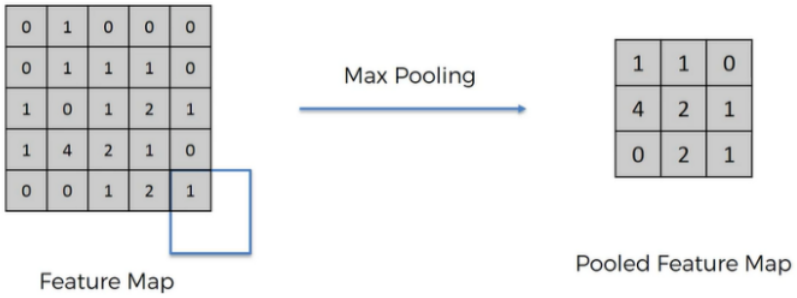


FIGURE 4.7 – *Exemple de Max Pooling*

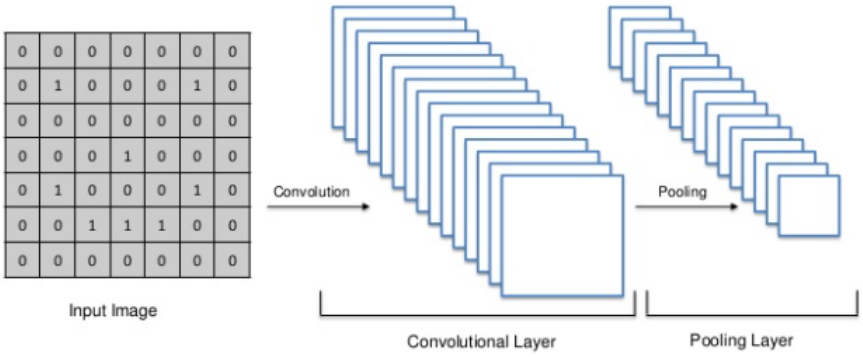


FIGURE 4.8 – *MaxPooling Process*

Les CNN s'appliquent donc sur des tableaux numériques, mais comment peut on appliquer ce genre de réseaux de neurones sur d'autre type de don-

nées ? Du texte par exemple ? Une solution possible intéressante consiste à ajouter une (ou plusieurs) couche d'intégration entre les données en entrée et la première couche du CNN.

4.4 Long Short Terme Memory(LSTM)

Les réseaux de mémoire à long terme et à court terme généralement appelés simplement (LSTM : Long Short Term Memory) sont un type spécial de RNN. Les Réseaux neuronaux récurrents présentés dans la section précédente sont capables d'apprendre des règles de mise à jour de séquence arbitraire en théorie. Dans la pratique, cependant, ces modèles oublient généralement rapidement le passé .
C'est ce qu'on appelle le problème de la disparition de gradient Hochreiter [1998] et c'est pourquoi ils ont inventé le LSTM.
La cellule LSTM est une adaptation de la couche récurrente qui permet aux signaux plus anciens des couches profondes de se déplacer vers la cellule du présent . La figure suivante représente une chaîne de trois cellules LSTM :

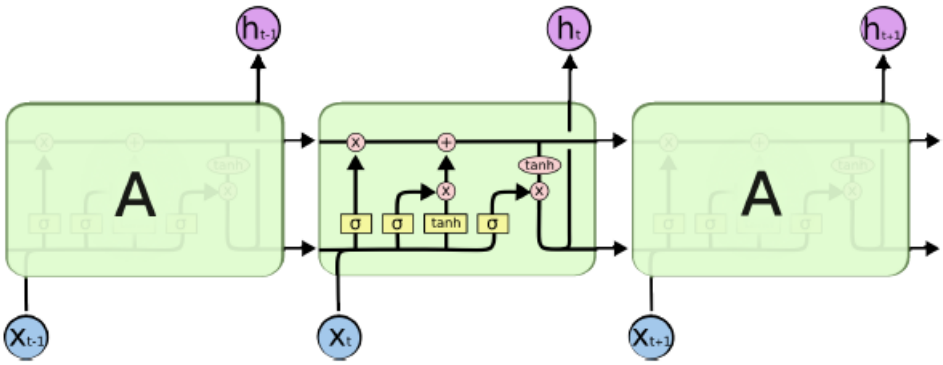
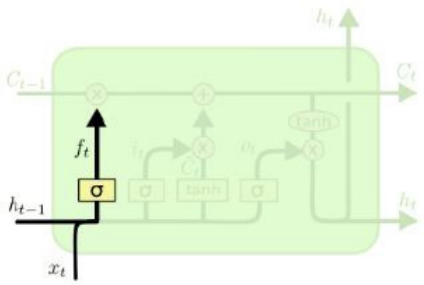


Figure 4.9 – *Une chaîne de cellules LSTM*

Les calculs se déroule comme suit [2] :

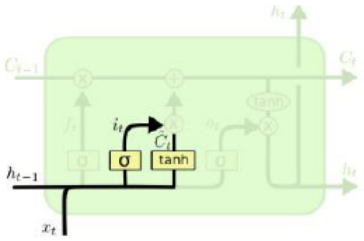


2. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Avec :

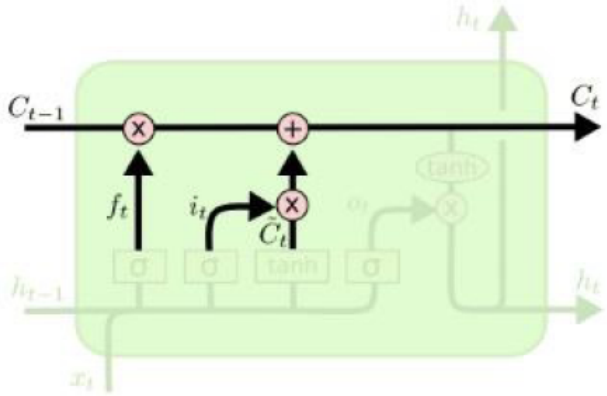
- h_{t-1} : La sortie a l'instant t-1
- x_t : L'entrée courant a l'instant t
- b : C'est le bais
- ω : C'est le poids
- σ : C'est la fonction sigmoïde



$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_C)$$

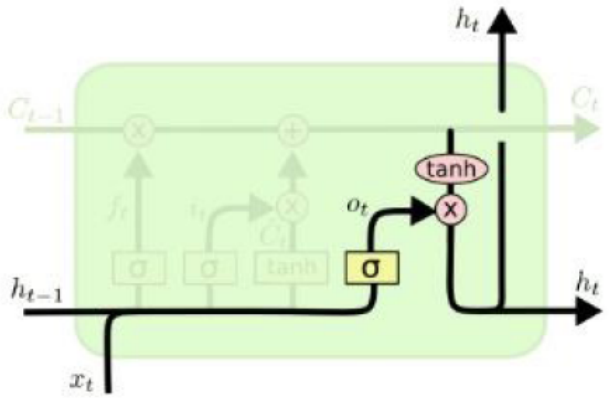
Avec :

- tanh : C'est la fonction d'activation tangente hyperbolique
- C_t : Une valeur candidate



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Avec :
c_t : État interne



$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
$h_t = o_t * tanh(C_t)$

Avec :
h_t : La sortie.

4.5 Word embedding

Le word embedding (« plongement de mots » ou « plongement lexical » en français) est une méthode d'apprentissage d'une représentation de mots utilisée notamment en traitement automatique des langues. Le terme devrait plutôt être rendu par vectorisation de mots pour correspondre plus proprement à cette méthode (Vukotić et al. [2015]).
Cette technique permet de représenter chaque mot d'un dictionnaire par un vecteur de nombres réels. Cette nouvelle représentation a ceci de particulier que les mots apparaissant dans des contextes similaires possèdent des vecteurs correspondants qui sont relativement proches. Par exemple, on pourrait s'attendre à ce que les mots « chien » et « chat » soient représentés par des vecteurs relativement peu distants dans l'espace vectoriel où sont définis ces vecteurs. Cette technique est basée sur l'hypothèse (dite « de Harris » ou distributional hypothesis) qui veut que les mots apparaissant dans des contextes similaires ont des significations apparentées [3].

4.5.1 Principe du plongement lexical de word2vec

L'idée générale est de projeter un ensemble de mots d'un vocabulaire de taille V dans un espace vectoriel continu où les vecteurs de ces mots ont une taille N relativement petite. De plus, on veut trouver une représentation vectorielle de chaque mot de V de façon que les mots aux représentations voisines apparaissent dans des contextes similaires [4].
L'approche de Mikolov et al. [2013] s'appuie sur des réseaux de neurones

3. https://bit.ly/2SR3xWT
4. https://bit.ly/3yJXUKp

artificiels pour construire ces vecteurs. Ces modèles sont entraînés sur des corpus très volumineux. Le principe de base est qu'on tente de prédire un mot à partir de son contexte ou vice-versa. Un sous-produit de cet apprentissage constitue les plongements lexicaux.
Il existe deux variantes de l'algorithme word2vec :

— Le modèle Continuous Bag-Of-Words (sac de mots continu) cherche à prédire un mot à partir de ses mots voisins, par **exemple** prédire attrape dans l'extrait le chat la souris.
— Le modèle skip-gram cherche à prédire les mots du contexte à partir d'un mot central, par exemple prédire les quatre mots le, chat, la et souris à partir de attrape.

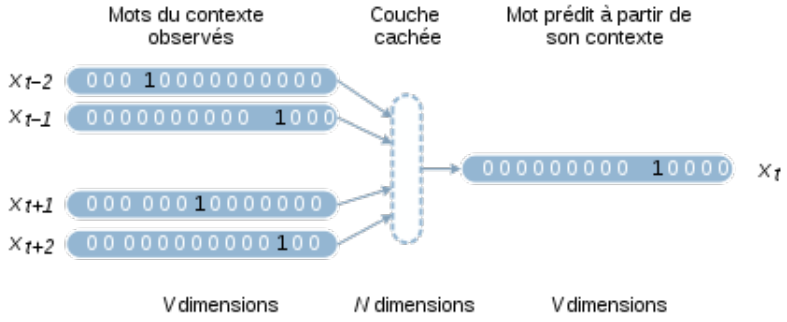


FIGURE 4.10 – *Modèle CBOW, Illustration adaptée de l'article de Mikolov et al. [2013]*

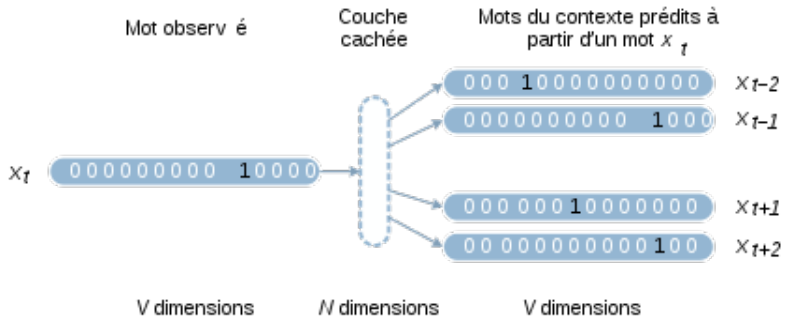


FIGURE 4.11 – *Modèle Skip-gram, Illustration adaptée de l'article de Mikolov et al. [2013]*

Continuous Bag-of-Words (CBOW)

Le modèle Continuous Bag-of-words est l'image inverse du skip-gram modèle. À partir d'un contexte $\{x_i, .., x_c\}$ il apprend à reconnaître un mot y_j, l'ensemble $\{x_i, .., x_c\} \bigcup \{y_j\}$ est une phrase du langage [5].

La couche cachée h est la somme pondérée de l'entrée X par W, on peut donc montrer que

$$h = \frac{1}{C} W . \sum_{i=1}^{C} x_i .$$

5. https://bit.ly/3yJXUKp

La couche de sortie Y peut être calculée sur la même idée, on a donc que l'entrée à la j^{ime} valeur du mot est : $u_j = v'^{T}_{w_j}.h$

Nous pouvons donc appliquer la fonction **softmax** qui nous donne

$$p(w_{y_j|w_1,..,w_C}) = y_j = \frac{\exp(u_j)}{\sum_{j=1}^{V} \exp(u'_j)}$$

La fonction de perte est définie comme étant $E = -\log p(w_O|w_I)$.

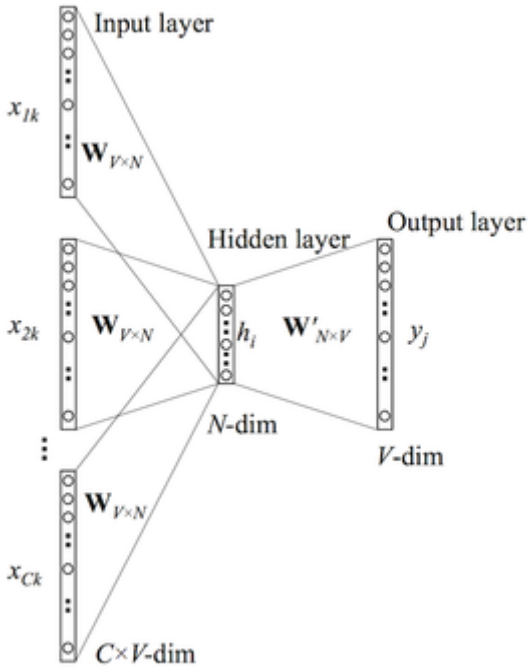


FIGURE 4.12 – *Réseau de neurones Continuous Bag-of-words*

4.6 Différentes couches utilisées

4.6.1 Flatten Layer

Cette couche convertit toutes les images filtrés (avec des caractéristiques différentes) en une liste. La raison pour laquelle on doit insérer cette couche, est que la couche suivante nécessite des données en entrée sous forme de liste. [6].

6. urlz.fr/9Upv

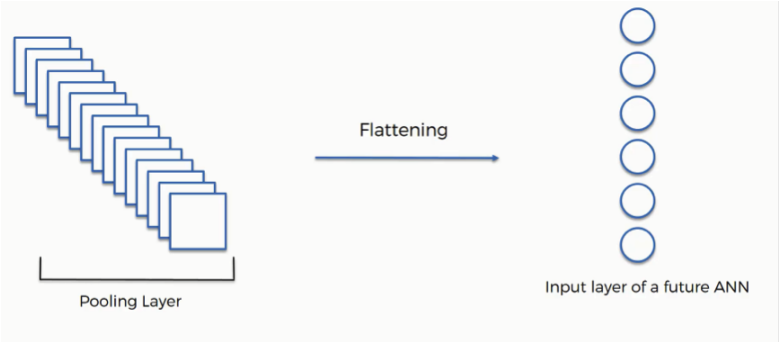


FIGURE 4.13 – *Le fonctionnement de Flatten Layer*

4.6.2 Dense Layer

Couche d'un réseau neuronal artificiel dans laquelle tous les nœuds contenus se connectent à tous les nœuds de la couche suivante.
Les couches entièrement connectée sont couramment utilisées dans les réseaux de neurones convolutifs et les réseaux de neurones récurrents. Une machine Boltzmann restreinte est un exemple de couche entièrement connectée.

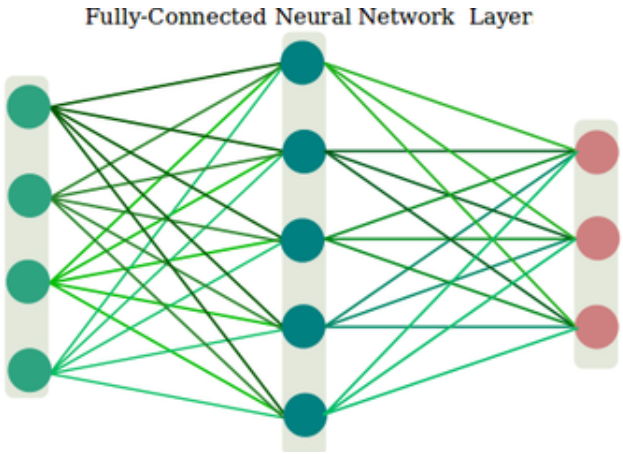


FIGURE 4.14 – *Couche entiérement connéctée*

4.6.3 Dropout Layer

Le décrochage, ou abandon, est une technique de régularisation pour réduire le surajustement dans les réseaux de neurones. La technique évite des co-adaptations complexes sur les données de l'échantillon d'entraînement. C'est un moyen très efficace d'exécuter un moyennage du modèle de calcul avec des réseaux de neurones. Le terme "décrochage" se réfère à une suppression temporaire de neurones (à la fois les neurones cachés et les neurones visibles) dans un réseau de neurones.

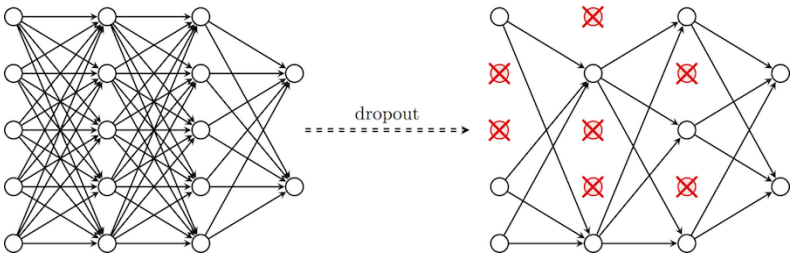


FIGURE 4.15 – *Schéma représentatif de la technique de dropout*

.

4.6.4 La couche de perte LOSS

Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées(La perte euclidienne ,Softmax , entropie croisée sigmoïde)

Implémentation 5

Pour implémenter notre réseau, il fallait choisir un langage puissant qui facilite la création des réseaux de neurones. Les deux langages de programmation majeurs utilisés pour le ML et le DL et dominants le marché sont Python et R. Alors il s'agissait de choisir l'un des deux langages selon nos besoins.

Après plusieurs recherches, on a décidé d'utiliser Python pour les raisons suivantes :

— Il est le plus utilisé . Cela conduit à une meilleure documentation et plus de discussion sur le web.
— Il fournit les meilleures bibliothèques dans ce domaine .

5.1 Matériel et environnement

5.1.1 La machine

Le projet a utilisé une seule machine pour mener les expériences.
Lenovo Thinkpad X240, Windows 10 édition professionnel , 64 bits installé. Avec un processeur Intel © $Core^{TM}$ i5-4300U CPU @ 1.90GHz 2.49 GHz. 8 Go de RAM.

5.1.2 Anaconda

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets conda.
La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS.

[1]

1. https://www.anaconda.com/what-is-anaconda

5.1.3 Jupyter Notebook

Jupyter Notebook , l'outil incontournable du data scientist , est une application Web Open Source permettant de créer et de partager des documents contenant du code (exécutable directement dans le document), des équations, des images et du texte. Avec cette application il est possible de faire du traitement de données, de la modélisation statistique, de la visualisation de données, du Machine Learning, etc. Elle est disponible par défaut dans la distribution Anaconda (suivre ce lien pour savoir comment l'installer)

[2].

5.1.4 Cloab

Google Colab ou Colaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

5.2 Bibliothèques utilisées :

5.2.1 Tensorflow

TensorFlow est une bibliothèque open source de Machine Learning, Créé par l'équipe Google Brain en 2011, sous la forme d'un système propriétaire dédié au réseaux de neurones de Deep Learning, TensorFlow s'appelait à l'origine DistBelief. Par la suite, le code source de DistBelief a été modifié et cet outil est devenu une bibliothèque basée application. En 2015, il a été renommé TensorFlow et Google l'a rendu open source. Depuis lors, il a subi plus de 21000 modifications par la communauté et est passé en version 1.0 en février 2017.


[3]

2. https://bit.ly/3cAvqZU
3. https://bit.ly/3vkfAt5

5.2.2 Keras

Keras est une bibliothèque open source de réseaux de neurones écrite en Python. Elle est capable de fonctionner sur TensorFlow, Microsoft Cognitive Toolkit, Theano ou PlaidML. Conçu pour permettre une expérimentation rapide avec des réseaux de neurones.
Elle offre un ensemble d'abstractions de niveau supérieur et plus intuitif facilitant le développement de modèles deep learning.

[4]

Puisque Keras contient la plupart des fonctionnalités de Tensorflow (qui est trop technique et bas niveau par rapport à Keras), généralement les modèles sont définis en utilisant Keras. De plus, puisque Keras repré- sente une couche supérieure de Tensorflow, on a toujours la possibilité de descendre à TensorFlow en cas de besoin d'une fonctionnalité spécifique de TensorFlow ou une fonctionnalité personnalisée que Keras ne prend pas en charge [5].
On a donc décidé d'utiliser Keras comme bibliothèque pour l'implémentation de notre modèle.

Caractéristiques de l'algorithme d'apprentissage :

1. **La fonction de perte (loss) :**l'erreur du modèle doit être estimée de manière répétée. Cela nécessite le choix d'une fonction d'erreur, généralement appelée fonction de perte (loss), qui peut être utilisée pour estimer l'erreur du modèle dans le but de faire un ajustement de poids qui améliorent le résultat de notre modèle.
   Plusieurs fonctions de calcul de perte existent dans Keras, on cite les plus utilisées (sachant que :y_i : la classe correcte, $\hat{y}_i$: la classe prédite) :
   — **Mean Squared Error :**utilisée généralement pour les problèmes de régression, calculée comme la moyenne des différences quadratiques entre les valeurs prédites et les valeurs réelles.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

   .
   — **Categorical Crossentropy :** la fonction de perte par défaut à utiliser pour les problèmes de classification multi-classes (une classification où les valeurs cibles sont dans l'ensemble {0, 1,2,3,. . . ,n}).

$$H(p,q) = E_p[-log(q)] = H(p) + D_{KL}(p \parallel q)$$

   Où p est la distribution correcte et q est la distribution estimée, est l'entropie de p, et $D_{KL}(p \parallel q)$ est la divergence de Kullback-Leibler entre q et p.

4. https://bit.ly/3cAwsFg
5. https://bit.ly/3zmQ2ig

— **Binary Crossentropy :** la fonction de perte par défaut à utiliser pour les problèmes de classification binaire, destinée à être utilisé avec une classification où les valeurs cibles sont dans l'ensemble$\{0, 1\}$.

$$-\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

2. **L'algorithme d'optimisation (optimizer) :**chaque fois qu'un réseau de neurones a fini un pass sur un lot de données, généré des résultats de prédiction et calculé l'erreur, il doit décider de la façon d'utiliser l'erreur pour ajuster les poids. L'algorithme qui détermine cette étape est appelée algorithme d'optimisation.
   — **Adaptive moment estimation (Adam) :** Adam (Kingma et Ba [2014]) est un optimiseur qui exploitent la puissance des méthodes de taux d'apprentissage adaptatifs (adaptive learning rates) pour trouver des taux d'apprentissage individuels pour chaque paramètre [6]. La mise à jour des poids se fait comme suite :

$$\omega_t = \omega_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

   Où ω_t est le nouveau poids .
   ω_{t-1} est l'ancien poids
   $\hat{m}_t$ et $\hat{v}_t$ sont des estimations corrigées du premier et du second moment du gradient est le step size.

   — **Root Mean Square Propagation(RMSProp) :**La racine de l'erreur quadratique moyenne (REQM) ou racine de l'écart quadratique moyen est une mesure fréquemment utilisée des différences entre les valeurs (valeurs d'échantillon ou de population) prédites par un modèle ou estimateur et les valeurs observées (ou vraies valeurs). La REQM représente la racine carrée du deuxième moment d'échantillonnage des différences entre les valeurs prédites et les valeurs observées.

   Ces écarts sont appelés résidus lorsque les calculs sont effectués sur l'échantillon de données qui a été utilisé pour l'estimation ou ils sont appelés erreurs (ou erreurs de prédiction) lorsqu'ils sont calculés sur des données hors de l'échantillon d'estimation.

   La REQM agrège les erreurs de prédiction de différents points de données en une seule mesure de puissance prédictive accrue. La REQM est une mesure de précision, qui sert à comparer les erreurs de différents modèles prédictifs pour un ensemble de données particulier et non entre différents ensembles de données, car elle dépend de l'échelle.

6. https://urlz.fr/9UuC

$$\text{REQM} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}.$$

La REQM des valeurs prédites $\hat{y}_t$ pour les instants t de la variable dépendante y_t d'une régression, avec des variables observées T fois, est calculée pour T différentes prédictions comme la racine carrée de la moyenne des carrés des écarts

3. **La métrique (metrics) :** Une métrique est une fonction utilisée pour juger la performance d'un modèle.
   L'utilisateur peut spécifier les métriques qu'il veut utiliser pour son modèle. Une fonction de métrique est similaire à une fonction de perte (peut être utilisée comme métrique aussi), sauf que les résultats de l'évaluation d'une métrique ne sont pas utilisées lors de l'entraînement du modèle mais enregistrés et affichés à l'utilisateur comme output.
   Parmi les métriques existantes dans Keras :
   — **Binary accuracy** : la moyenne des cas où la classe prédite est égale à la classe réelle : K.mean(K.equal(y_true,K.round(y_pred))) K.round (y_pred) implique que le seuil est 0.5, tout ce qui est supérieur à 0.5 sera considéré comme correcte.

Paramètres du réseau :

— **Epochs :** : une époque est lorsque le dataset en entier est passé en avant et en arrière (forward et back propagation) à travers le réseau de neurones une seule fois. Par contre, il faut passer le dataset plusieurs fois dans le même réseau, puisque le processus d'ajustement de poids est itératif et une fois ne suffit jamais pour avoir les meilleurs ajustements. Il n'existe pas une méthode pour définir le nombre d'époques car ce nombre dépend de la diversité des données utilisées.

Entraîner le réseau de neurones sur une ou deux époques engendre un **underfitting** (le sous-apprentissage fait référence à un modèle qui ne peut ni modéliser les données d'apprentissage ni se généraliser à de nouvelles données).
De même, l'entraîner sur un grand nombre d'époques cause un **overfitting** (un surapprentissage survient lorsqu'un modèle apprend les détails des données d'apprentissage dans la mesure où ils ont un impact négatif sur les performances du modèle avec de nouvelles données).
La solution est d'essayer des nombres d'époques différentes puis choisir le meilleur pour nos données.

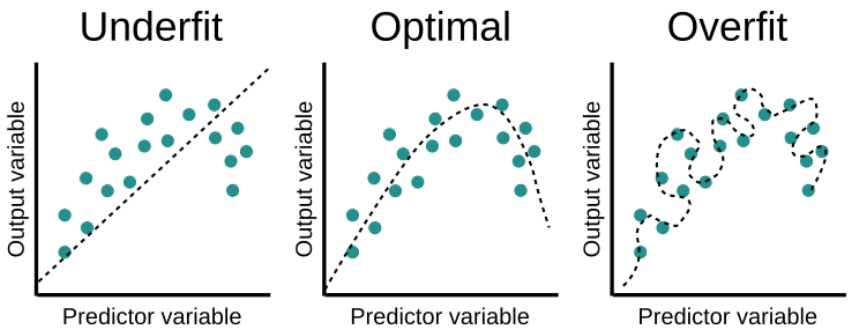


FIGURE 5.1 – *L'impact du choix du nombre d'époques sur les résultats.*

— **Batch size :** Le batch size est la taille d'un lot de données, puisqu'un ordinateur ne peut pas charger toutes les données du dataset pour les utiliser, on divise généralement le dataset en batch.
Faire le training pour tous les batchs une seul fois représente une époque.

5.2.3 Pandas

Pandas est une librairie python qui permet de manipuler facilement des données à analyser par laquelle nous pouvons :

— Manipuler des tableaux de données avec des étiquettes de variables (colonnes) et d'individus (lignes).
— Ces tableaux sont appelés DataFrames, similaires aux dataframes sous R.
— On peut facilement lire et écrire ces dataframes à partir ou vers un fichier tabulé.
— Tracer facilement des graphes à partir de ces DataFrames grâce à matplotlib [7].

5.2.4 Numpy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

[8]

7. https://bit.ly/34GIQiE
8. https://bit.ly/3ghVviT

5.2.5 Matplotlib

est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy

5.2.6 Gensim

Gensim est une bibliothèque de modélisation de sujets pour Python qui donne accès à Word2Vec et à d'autres algorithmes d'intégration de mots pour la formation, et permet également de charger des intégrations de mots pré-entraînées que vous pouvez télécharger sur Internet.

5.3 L'architectures de notre approche

Après avoir eu idée sur les modèles qui ont été proposés pour détecter le hate speech dans le 3éme chapitre , Faris et al. [2020] et Mohaouchane et al. [2019] ont fait une hybridation de CNN-LSTM , ils nous ont inspiré pour faire cette hybridation donc nous avons choisi de faire une hybridation de CNN-BiLSTM.

5.3.1 Un réseau hybride :

Un réseau hybride est simplement l'enchaînement de deux réseaux. On utilise la notion "hybride" (ou on les relie par un tiret) pour souligner que leur travail est complémentaire/intriqué. Par exemple pour le CNN-LSTM, on peut considérer que le CNN est la phase de préparation des données pour le LSTM. Cette phase est "adaptative"/"intelligente", contrairement à un pré-traitement plus classique (avec des outils mathématiques/statistiques), puisque le CNN travaillera chaque donnée différemment en fonction de son contenu.

Couches du modèle hybride

Notre réseau se compose principalement de :
— Input Layer
— Embedding Layer(avec AraVec)
— Convolutional Layer (Conv1D)
— Max Pooling Layer
— BiLSTM Layer
— Flatten Layer
— Fully connected Layer
— Dropout Layer
— Output Layer (activation='sigmoïd')

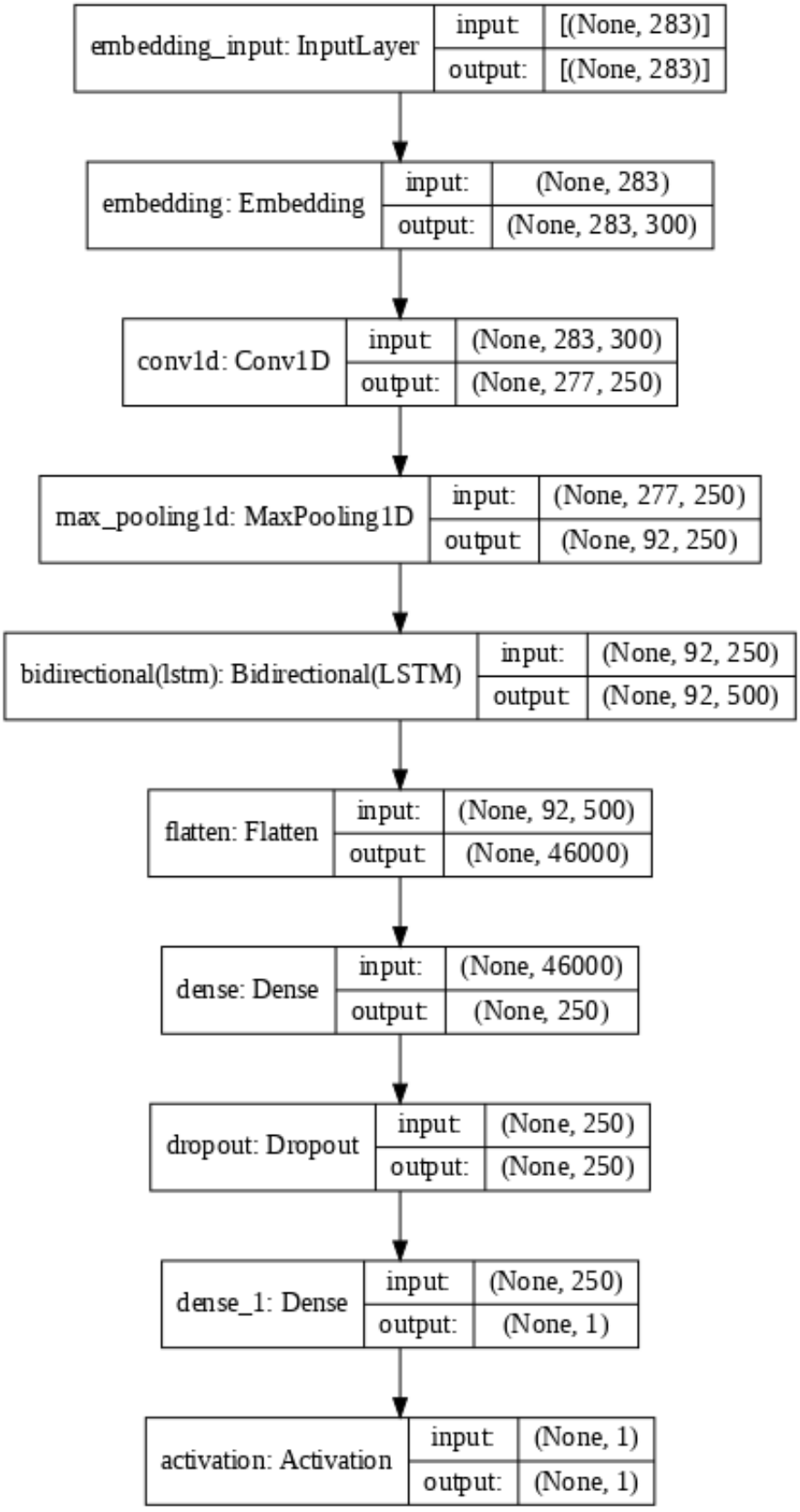


FIGURE 5.2 – *Hybridation CNN-BiLSTM, Model1*

5.3.2 Un réseau BiLSM

Le réseau LSTM (BiLSTM) a la capacité d'extraire les informations contextuelles des séquences de caractéristiques en traitant à la fois les dépendances avant et arrière.

Couches du modéle BiLSTM

— Input Layer
— Embedding Layer(avec AraVec)
— BiLSTM Layer
— Flatten Layer
— Fully connected Layer
— Dropout Layer
— Output Layer (activation='sigmoïd')

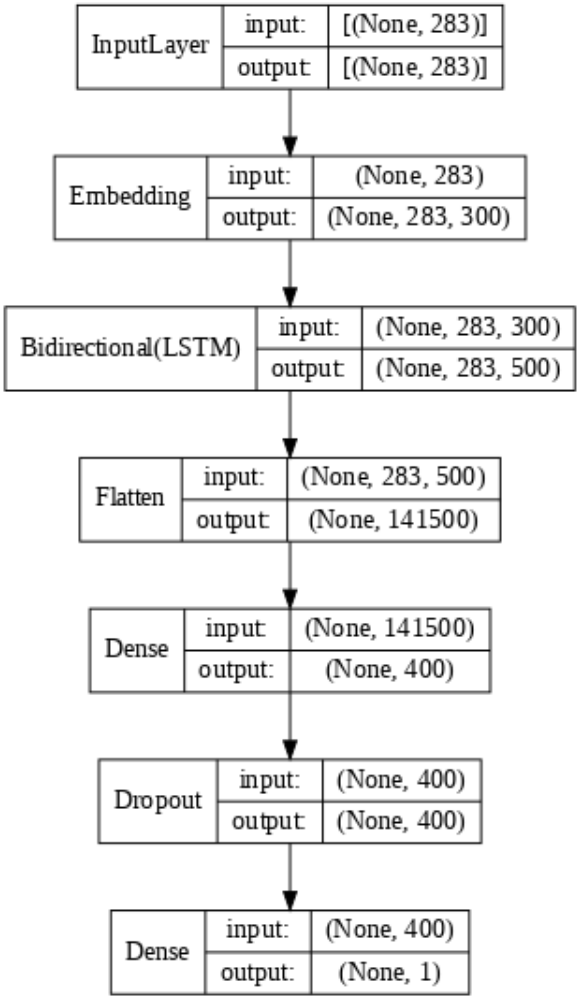


FIGURE 5.3 – *BiLSTM, Model2*

5.3.3 Embedding Layer

Dans un premier temps nous avons utilisé une couche d'intégration sans l'avoir initialiser , donc l'initialisation se faisait aléatoirement ce qui a influé sur les résultats (les résultats changent d'une exécution à l'autre).
Pour remédier à ce problème nous avons décidé d'utiliser des poids des mots fixe et comme nous nous intéressons au textes arabes , Aravec est le seul qui nous donne cette représentation unifiée que nous cherchions .

AraVec est un projet fait par Soliman et al. [2017] open source de représentation de mots distribués (incorporations de mots) qui vise à fournir à la communauté de recherche en NLP arabe des modèles d'incorporation de mots puissants et gratuits. Les modèles présentés ont été construits avec soin en utilisant plusieurs ressources de texte arabe différentes pour fournir une large couverture de domaine.
Plus précisément, les modèles ont été construits à l'aide de pages Web collectées sur le World Wide Web, de texte récolté à partir de plateformes sociales(tel que Twitter) et de texte obtenu à partir d'entrées d'encyclopédie.

Pour notre cas on est en Social Media donc nous avons choisi Twitter avec représentation Word2Vec CBOW avec un vocabulaire qui contient 1,476,715 mot et qui va même nous aider pour avoir la similitude entre les mots .

```
egypt = nlp("مصر")
tunisia = nlp("تونس")
apple = nlp("تفاح")

print("egypt Vs. tunisia = ", egypt.similarity(tunisia))
print("egypt Vs. apple = ", egypt.similarity(apple))
```

```
egypt Vs. tunisia =  0.8277813628455393
egypt Vs. apple =  0.09689554003631644
```

FIGURE 5.4 – *Exemple AraVec similarity*

Pour l'intégrer à notre code , nous sommes obligés d'utiliser Gensim .
Les CNN et LSTM prennent en entrée une taille fixe des données et puisque les phrases n'ont pas la même longueur, il fallait faire un padding en ajoutant des zéros à la fin('post') de la représentation de la phrase jusqu'à atteindre la taille souhaitée.

```
from tensorflow.keras.preprocessing.text import one_hot, Tokenizer
max_num_words = 25354
max_length =40

tokenizer = Tokenizer(num_words=max_num_words)
tokenizer.fit_on_texts(list(X_train))

list_tokenized_train = tokenizer.texts_to_sequences(X_train)
list_tokenized_test = tokenizer.texts_to_sequences(X_val)

list_tokenized_train[1]
```

```
[918, 918, 33, 262, 918]
```

```
X_tr = pad_sequences(list_tokenized_train, maxlen=max_length, padding='post')
X_tes = pad_sequences(list_tokenized_test, maxlen=max_length, padding='post')
X_tr[1]
```

```
array([918, 918,  33, 262, 918,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0], dtype=int32)
```

FIGURE 5.5 – *Avant - Après utilisation de padding*

Après avoir terminer avec le padding, nous allons initialiser en utilisant Aravec pour avoir une représentation unique pour chaque mot .

```
unique_words = len(word_index)
total_words = unique_words+1
skipped_words = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
  try:
        embedding_vector = model[word]
  except:
    skipped_words = skipped_words+1
    pass
  if embedding_vector is not None:
    embedding_matrix[index] = embedding_vector
print("Embeddings Matrix shape : ",embedding_matrix.shape)
```

FIGURE 5.6 – *Initialisation des poids avec Aravec*

Pour s'assurer que la représentation selon AraVec a été bien faite ,nous allons comparé un mot dans notre matrice que nous avons obtenu sa représentation (embedding_matrix) avec la représentation dans le modèle pré-entrainé AraVec , et après avoir fait une soustraction entre les deux on remarque que la matrice obtenu est égale a o donc , ils ont les mêmes poids donc la même représentation.

```
embedding_matrix[27498]-model.wv['تمس']
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

FIGURE 5.7 – *Initialisation avec Aravec*

Passons maintenant à l'initialisation de la couche d'embedding :

```
from tensorflow.keras.layers import *
# create the embedding layer
embedding_layer = Embedding(total_words,
                            embedding_dim,
                            weights=[embedding_matrix],
                            input_length=max_length)
```

FIGURE 5.8 – *Initialisation de la couche d'intégration*

— Total_words : représente la taille du vocabulaire (le nombre de mots détectés par le tokenizer)
— Embedding_dim : représente la taille des vecteurs de sortie de cette couche.
— Input_length : représente la longueur des séquences d'entrée pour notre cas.
— Weights : les poids utilisés pour cette couche, si ce paramètre n'est pas précisé ils seront initialisés aléatoirement.

5.3.4 Construction des deux modèles :

Pour construire notre modèle nous allons utilisé *Sequential()* de Keras .

Premier modèle :

— La couche de convolution prend en paramètre :filters=250 ;kernel_size=7 ; strides=1 ; padding="valid" et relu en fonction d'activation
— MaxPooling prend en paramètre 3
— BiLSTM composé de 250 units ou neurones.
— Fully Connected layer prend 250 units avec 'relu' pour l'activation
— Dropout prend le *rate* = 0.3
— Au final en a Output Layer avec activation sigmoïd car nous avons que deux classe.

```
model3= Sequential()
model3.add(embedding_layer )
model3.add(Conv1D(250, 7,activation='relu',padding="valid", strides=1))

model3.add(MaxPooling1D(pool_size=3))
model3.add(Bidirectional(LSTM(units=250,return_sequences=True)))
model3.add(Flatten())
model3.add(Dense(250, activation='relu'))
model3.add(Dropout(0.3))
model3.add(Dense(1))
model3.add(Activation('sigmoid'))
```

FIGURE 5.9 – *Construction du premier modèle*

Deuxième modèle :

— BiLSTM composé de 250 units ou neurones.
— Fully Connected layer prend 250 units avec 'relu' pour l'activation
— Dropout prend le *rate* = 0.2
— Au final en a Output Layer avec activation sigmoïd car nous avons que deux classe.

```
model2=Sequential()
model2.add(embedding_layer )

model2.add(Bidirectional( LSTM(units=250,
                               return_sequences=True)))

model2.add(Flatten())
#Fully connected layer
model2.add(Dense(400, activation='relu'))
#Drop out layer
model2.add(Dropout(0.2))
#Output  Layer
model2.add(Dense(1, activation='sigmoid'))
```

FIGURE 5.10 – *Construction du deuxième modèle*

5.4 Apprentissage et évaluation

Pour évaluer les performances des réseaux, nous avons utilisé quatre métriques bien connues,à savoir, l'accuracy, la précision, le rappel et F1-mesure ou F1-score . Il est largement utilisé pour mesurer la prédiction des Hate Speech
La précision, le rappel et la F1-mesure sont définis comme suit :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 - measure = \frac{2.precision}{precision + recall}$$

Pour évaluer notre modèle ,nous avons consacré 20% des données pour le test tout en faisant un désordonnement (shuffle) et cela est fait grâce a *train_test_split* de la bibliothèque *sklearn*.

5.4.1 Premier dataset de Mubarak et al. [2017]

Après compilation du premier modèle (Hybridation) avec RMSprop comme optimisateur,loss binary crossentropy et l'accuracy comme mesure .
Le fit est fait en utilisant 15 epochs et batch_size à 32 , verbose=1 ; la figure 5.11 nous montre le résultat d'apprentissage .

```
793/793 [==============================] - 77s 98ms/step - loss: 0.3191 - accuracy: 0.8666
Epoch 4/15
793/793 [==============================] - 77s 98ms/step - loss: 0.2539 - accuracy: 0.8966
Epoch 5/15
793/793 [==============================] - 78s 98ms/step - loss: 0.1810 - accuracy: 0.9298
Epoch 6/15
793/793 [==============================] - 78s 98ms/step - loss: 0.1314 - accuracy: 0.9499
Epoch 7/15
793/793 [==============================] - 78s 98ms/step - loss: 0.0915 - accuracy: 0.9666
Epoch 8/15
793/793 [==============================] - 77s 97ms/step - loss: 0.0728 - accuracy: 0.9764
Epoch 9/15
793/793 [==============================] - 77s 98ms/step - loss: 0.0598 - accuracy: 0.9803
Epoch 10/15
793/793 [==============================] - 78s 98ms/step - loss: 0.0476 - accuracy: 0.9848
Epoch 11/15
793/793 [==============================] - 78s 98ms/step - loss: 0.0424 - accuracy: 0.9866
Epoch 12/15
793/793 [==============================] - 77s 98ms/step - loss: 0.0406 - accuracy: 0.9871
Epoch 13/15
793/793 [==============================] - 77s 97ms/step - loss: 0.0365 - accuracy: 0.9891
Epoch 14/15
793/793 [==============================] - 77s 97ms/step - loss: 0.0368 - accuracy: 0.9901
Epoch 15/15
793/793 [==============================] - 77s 97ms/step - loss: 0.0353 - accuracy: 0.9903
```

Figure 5.11 – *Apprentissage de l'expérimentation 1*

Nous remarquons que notre modèle apprend si vite et que l'accuracy de

la 15^{ieme} est a 99% , ainsi que la valeur loss la plus minimale est a 0.035.
Le graphe suivant nous montre l'évolution de loss et d'accuracy durant l'apprentissage de notre modèle selon chaque epochs .

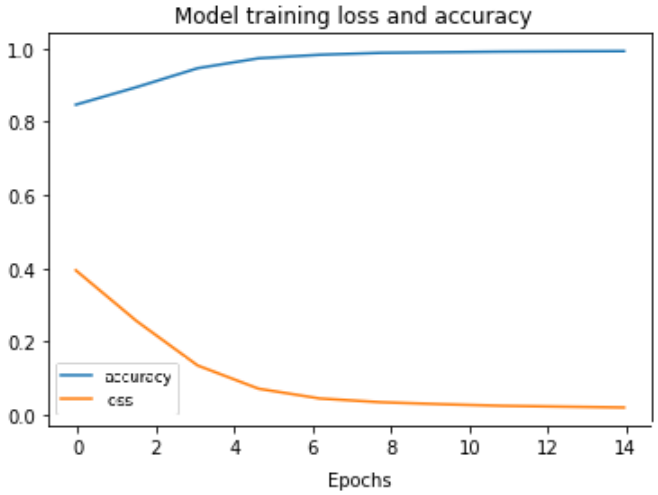


FIGURE 5.12 – *Graphe de "accuracy" et "loss" suivant chaque epochs*

En suite après évaluation nous avons obtenu l'accuracy a 81% avec meilleur loss 0.3923.

PERFORMANCE

Accuracy: 0.8068

	precision	recall	f1-score	support
0	0.44	0.34	0.38	1119
1	0.86	0.91	0.89	5220
accuracy			0.81	6339
macro avg	0.65	0.62	0.63	6339
weighted avg	0.79	0.81	0.80	6339

FIGURE 5.13 – *Résultats de l'expérimentation 1*

Passons maintenant au deuxième modèle BiLSTM , il a été compilé comme le modèle précédant (RMSprop(lr=0.001),binary_crossentropie et accuracy).
Pour le fit , nous avons utilisé seulement 10 epochs et le batch_size a 32.

```
Epoch 1/10
793/793 [==============================] - 81s 99ms/step - loss: 0.3945 - accuracy: 0.8465
Epoch 2/10
793/793 [==============================] - 77s 98ms/step - loss: 0.2557 - accuracy: 0.8946
Epoch 3/10
793/793 [==============================] - 77s 97ms/step - loss: 0.1347 - accuracy: 0.9461
Epoch 4/10
793/793 [==============================] - 77s 97ms/step - loss: 0.0714 - accuracy: 0.9732
Epoch 5/10
793/793 [==============================] - 78s 98ms/step - loss: 0.0447 - accuracy: 0.9828
Epoch 6/10
793/793 [==============================] - 77s 97ms/step - loss: 0.0345 - accuracy: 0.9881
Epoch 7/10
793/793 [==============================] - 77s 97ms/step - loss: 0.0290 - accuracy: 0.9897
Epoch 8/10
793/793 [==============================] - 77s 98ms/step - loss: 0.0247 - accuracy: 0.9914
Epoch 9/10
793/793 [==============================] - 77s 97ms/step - loss: 0.0223 - accuracy: 0.9922
Epoch 10/10
793/793 [==============================] - 77s 97ms/step - loss: 0.0201 - accuracy: 0.9928
```

FIGURE 5.14 – *Apprentissage de l'expérimentation 2*

Passons maintenant à l'évaluation : nous avons obtenu 79% d'accuracy , la figure qui suit englobe les résultats avec les différentes mesures.

PERFORMANCE

Accuracy: 0.7916

	precision	recall	f1-score	support
0	0.42	0.46	0.44	1119
1	0.88	0.86	0.87	5220
accuracy			0.79	6339
macro avg	0.65	0.66	0.66	6339
weighted avg	0.80	0.79	0.80	6339

FIGURE 5.15 – *Résultats de l'expérimentation 2*

Pour voir l'influence du choix du nombre d'epochs ;nous avons lancé une autre fois le modèle avec epochs =5 en gardant toujours les mêmes paramètres . Nous avons remarqué une détérioration de performances par rapport a celles obtenues avant (Avant : accuracy=79 ;16%, Aprés :78,72) et surtout une trés grande valeur de loss (loss=3,23).
Cela nous confirme qu'un nombre élevé d'époques peut causer un overfitting (expliqué en 5.2.2).

PERFORMANCE

Accuracy: 0.7872

	precision	recall	f1-score	support
0	0.41	0.49	0.45	1119
1	0.89	0.85	0.87	5220
accuracy			0.79	6339
macro avg	0.65	0.67	0.66	6339
weighted avg	0.80	0.79	0.79	6339

FIGURE 5.16 – *Résultats de l'expérimentation 3*

5.4.2 Deuxième dataset de Alakrot et al. [2018]

Afin de confirmer le bon fonctionnement et rendement de nos modèles nous avons décidé de les tester sur ce deuxième dataset de Alakrot et al. [2018].
Premièrement nous avons ajouté de nouveaux pré-traitement tel que la normalisation de quelques lettres qui ont été proposés par Soliman et al. [2017] pour avoir une meilleur représentation de mots.

```
# Clean/Normalize Arabic Text
def clean_str(text):
    search = ["أ","¡","إ","ة","_","-","/",".","،"," و "," يا ",'"',"-","'","ى","\\",'\n', '\t','&quot;','?','؟','!']
    replace = ["ا","ا","ا","ه"," "," ","","","","و","يا","","","","ي","",' ', ' ',' ',' ? ',' ؟ ',' ! ']
```

FIGURE 5.17 – *Normalisation de mots*

Passons maintenant à l'apprentissage ; le premier modèle a été compilé avec RMSprop (lr=0.001),binary_crossentropy pour loss et accuracy comme mesure .
Ensuite nous avons utilisé des callbacks qui sont des fonctions qui s'appliquent à un moment donné de la procédure d'entraînement (fournit par Keras).
Parmi les fonctions de callback nous utilisons :

- CSVLogger qui nous permet de stocker les résultats obtenu dans un fichier de type .log (Facilite le plot des graphe).
- EarlyStopping arrête l'entraînement du modèle quand la "val_accuracy" n'est pas maximale par rapport à ses précédentes.
- ModelCheckpoint est utilisé pour sauvegarder le meilleur modèle.

Le tableau de callbacks est passé comme l'argument callbacks de la méthode d'entraînement model.fit ; avec 10 epochs et 32 en batch_size.

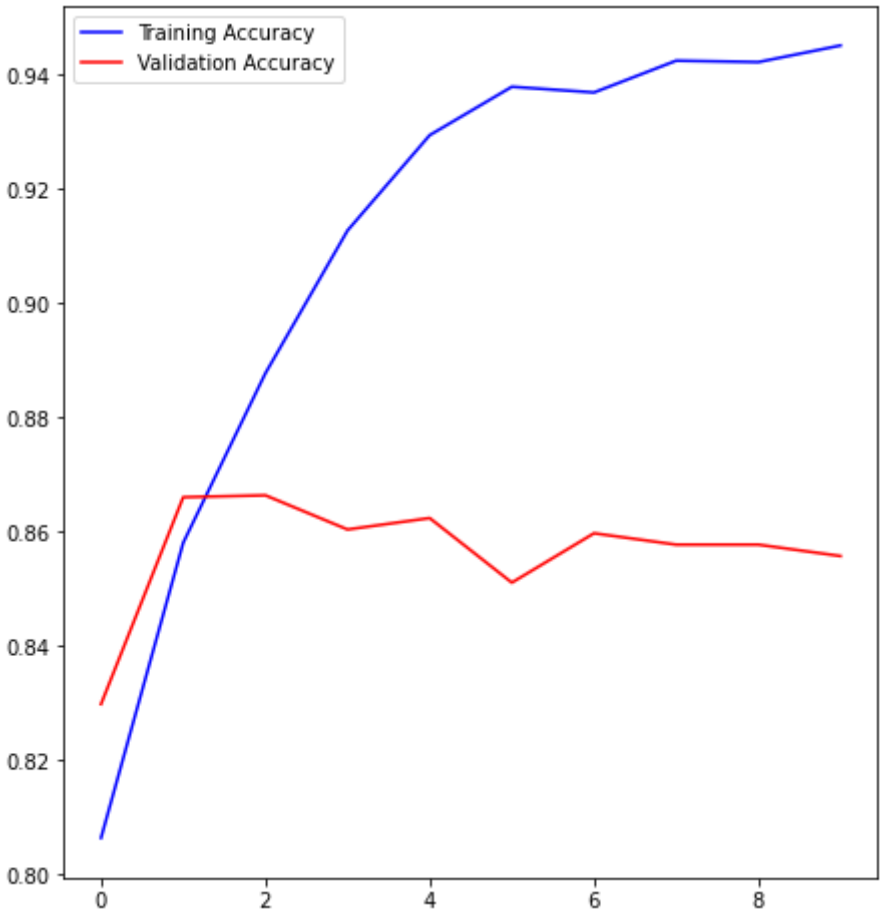


FIGURE 5.18 – *Comparaison d'acuuracy de quatrième expérimentation selon chaque epoch.*

L'écart entre le graphe du train et celui du test n'est pas grand cela signifie que l'apprentissage est bon.
Le EarlyStopping a marqué que durant la troisième epochs nous avons eu la meilleure $Accuracy = 0.86645$.
Aprés avoir terminer toutes les epochs nous avons trouvé que toutes les performances sont bonnes sauf pour le rappel de la classe 1 qui a 0.49 qui est probablement affecté par le nombre minime(672) par rapport a l'autre classe.

PERFORMANCE

Accuracy: 0.8558

	precision	recall	f1-score	support
0	0.87	0.96	0.91	2338
1	0.78	0.49	0.60	672
accuracy			0.86	3010
macro avg	0.83	0.73	0.76	3010
weighted avg	0.85	0.86	0.84	3010

FIGURE 5.19 – *Résultats de la quatrième expérimentation*

Adam a été utilisé pour compiler notre modèle avec un batch_size a 64 et 10 epochs pour le fit sinon tous les autres paramètres sont restés les mêmes , nous nous sommes arrivés a améliorer f-mesure par rapport a l'ancienne valeur de 0,1 .

	precision	recall	f1-score	support
0	0.87	0.96	0.91	2338
1	0.77	0.52	0.62	672
accuracy			0.86	3010
macro avg	0.82	0.74	0.77	3010
weighted avg	0.85	0.86	0.85	3010

FIGURE 5.20 – *Résultats de la cinquième expérimentation*

Passons maintenant au deuxième modèle qui a été compilé avec , Adam(learning_rate=0.001),binary_crossentropy,accuracy
Pour le fit nous avons utilisé EarlyStopping ,CSVLogger ainsi qu'un nombre d'epochs a 10 et batch_size=64.
Les résultats se sont amélioré par rapport aux résultats trouvés par le premier modéle , meilleur accuracy avec EarlyStopping était à 87,04% et meilleur loss=0.3493 ; et en passant par toues les epochs nous avons trouvé les résultats qui suivent 5.21.

PERFORMANCE

Accuracy: 0.8688

	precision	recall	f1-score	support
0	0.88	0.96	0.92	2338
1	0.81	0.54	0.65	672
accuracy			0.87	3010
macro avg	0.84	0.75	0.78	3010
weighted avg	0.86	0.87	0.86	3010

FIGURE 5.21 – *Résultats de la sixième expérimentation*

Le graphe qui suit nous montre que l'écart entre le loss et val_loss n'est pas très grand (loss=0,2 et val_loss=0,4) ainsi que pour l'accuracy (accuracy=0.919 , val_accuracy=0.868) et qui signifie que notre modèle est bon .

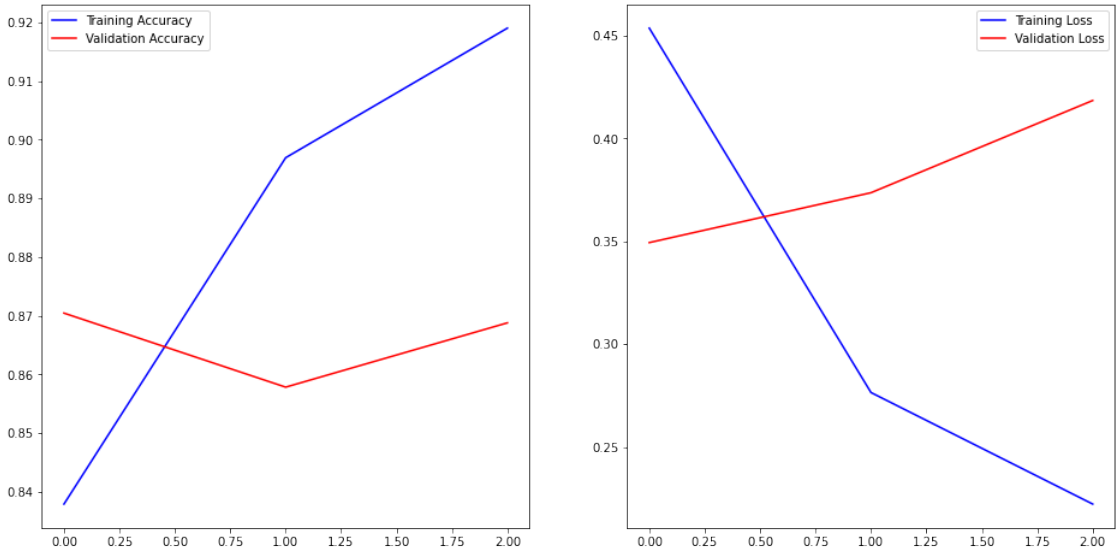


FIGURE 5.22 – *Graphe "loss" et "accuracy" de la sixième expérimentation selon les epochs*

Essayons maintenant d'effectuer un changement sur les paramètres du premier modèle , ou le kernal_size qui a été 7 est remplacé par 3 .
Nous l'avons compilé avec Adam(lr=0.01), binary_crossentropy et accuracy. Pour le fit nous utilisant 15 epochs et batch_size a 64 et aussi un callbacks , nous remarquons que l'accuracy du test a stagné à 0.7767 et il y eu une détérioration de précision, rappel et f-mesure(0.6 ;0.78 ;0.68 respectivement) ; ainsi que pour son loss qui n'a pas reconnu un grand changement et ce qui est montré par le graphe qui suit 5.23 donc la réduction de kernal_size n'est pas vraiment une bonne idée pour ce cas .

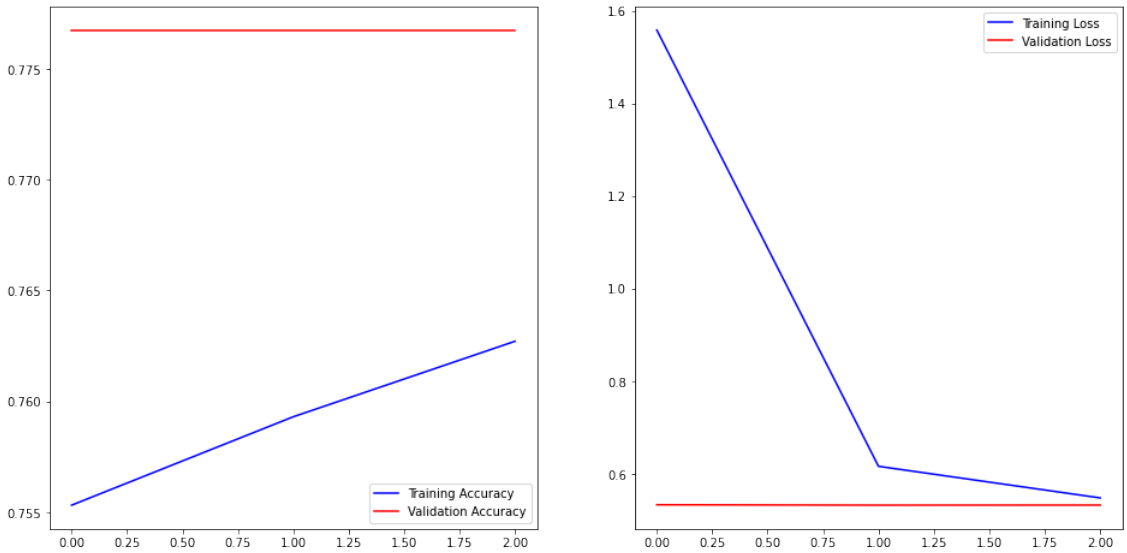


FIGURE 5.23 – *Graphe "loss" et "accuracy" de la septième expérimentation selon les epochs*

5.4.3 Troisième dataset

Un autre pré-traitement a été ajouté pour ce dataset ;nous avons choisi de supprimer les mots vides arabe et cela est fait a l'aide de la bibliothèque NLTK(Natural Language Toolkit) qui contient tous les stop words de différentes langues y compris l'arabe.
Au début nous avons laissé les paramètres des deux modèles tels qu'ils sont pour entraîner et évaluer les modèles mais ce tunning a nuit au performance , nous avons supposé que le nombre des paramètres est très grand pour un tel dataset qui contient que 6938 instances, pour confirmer ou infirmer ça nous avons réduit les nombres des paramètres des deux modéles .

Concernant BiLSTM : le nombres de units est à 128 ainsi que pour fully connected layer ou le nombre de units est à 100.
Puis nous compilons avec Adam(learning_rate=0.001), binary_crossentropy, le fit est effectué avec batch_size=16 ,10 epochs , verbose=2 et aussi EarlyStopping , nous avons obtenu une meilleure accuracy a 0.8393(mais une valeur de loss très élevée ; loss=2,8426).

PERFORMANCE

	precision	recall	f1-score	support
0	0.85	0.88	0.87	846
1	0.80	0.76	0.78	542
accuracy			0.83	1388
macro avg	0.83	0.82	0.82	1388
weighted avg	0.83	0.83	0.83	1388

FIGURE 5.24 – *Résultats de la huitième expérimentation*

Pour l'hybridation de CNN-BiLSTM , le filtre de CNN est à 128 , le kernel_size est à 3 , le BiLSTM a 128 units ainsi que pour fully connected layer . Nous compilons avec l'optimisateur 'adam' , loss 'binary_crossentropy' ; ensuite nous effectuons un fit avec 100 epochs ,batch_size a 32 , verbose=1 .

Après avoir terminer le tout, nous remarquons d'après la figure 5.25 que l'accuracy des deux graphes sont sur la même longueur d'onde ,ce qui prouve que le modèle est bon ainsi que la configuration des paramètres , et donc un un nombre de paramètres plus réduit pour ce dataset convient mieux ,la figure 5.26 englobe les différents résultats obtenus .

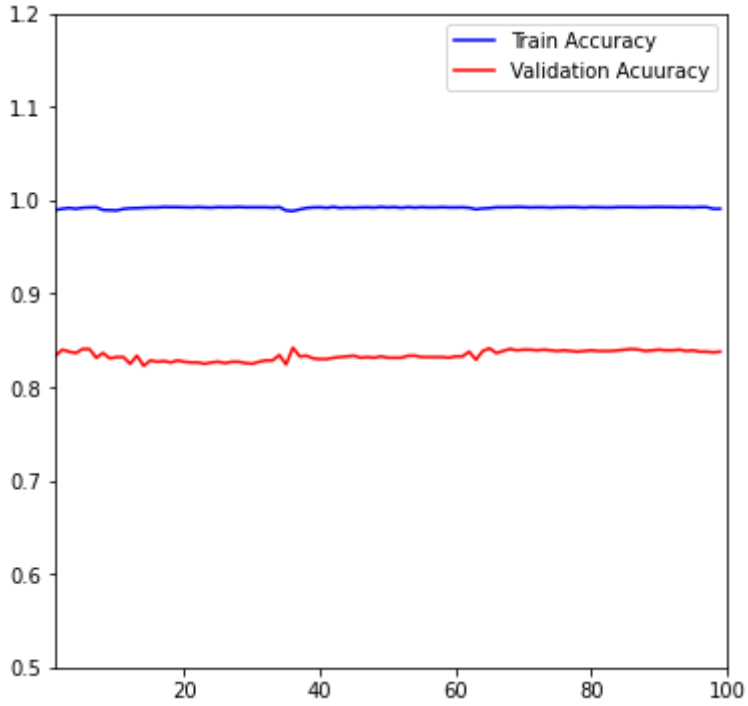


FIGURE 5.25 – *Graphe "Accuracy" de la neuvième expérimentation selon les epochs*

PERFORMANCE

Accuracy: 0.8379

	precision	recall	f1-score	support
0	0.85	0.89	0.87	846
1	0.81	0.76	0.79	542
accuracy			0.84	1388
macro avg	0.83	0.82	0.83	1388
weighted avg	0.84	0.84	0.84	1388

FIGURE 5.26 – *Résultats de la neuvième expérimentation*

Une autre configuration a été proposée pour ce modèle : filter =250 , kernel_size=5, 250 units pour BiLSTM Layer ainsi pour fully connected layer. Nous avons effectué la même compilation que la précédente, pour le fit nous avons mis 20 epochs et batch_size=32.
La meilleure accuracy obtenu est à 0.8429 avec meilleure loss 0.5535 .

PERFORMANCE

	precision	recall	f1-score	support
0	0.86	0.88	0.87	846
1	0.81	0.77	0.79	542
accuracy			0.84	1388
macro avg	0.83	0.83	0.83	1388
weighted avg	0.84	0.84	0.84	1388

FIGURE 5.27 – *Résultats de la dixième expérimentation*

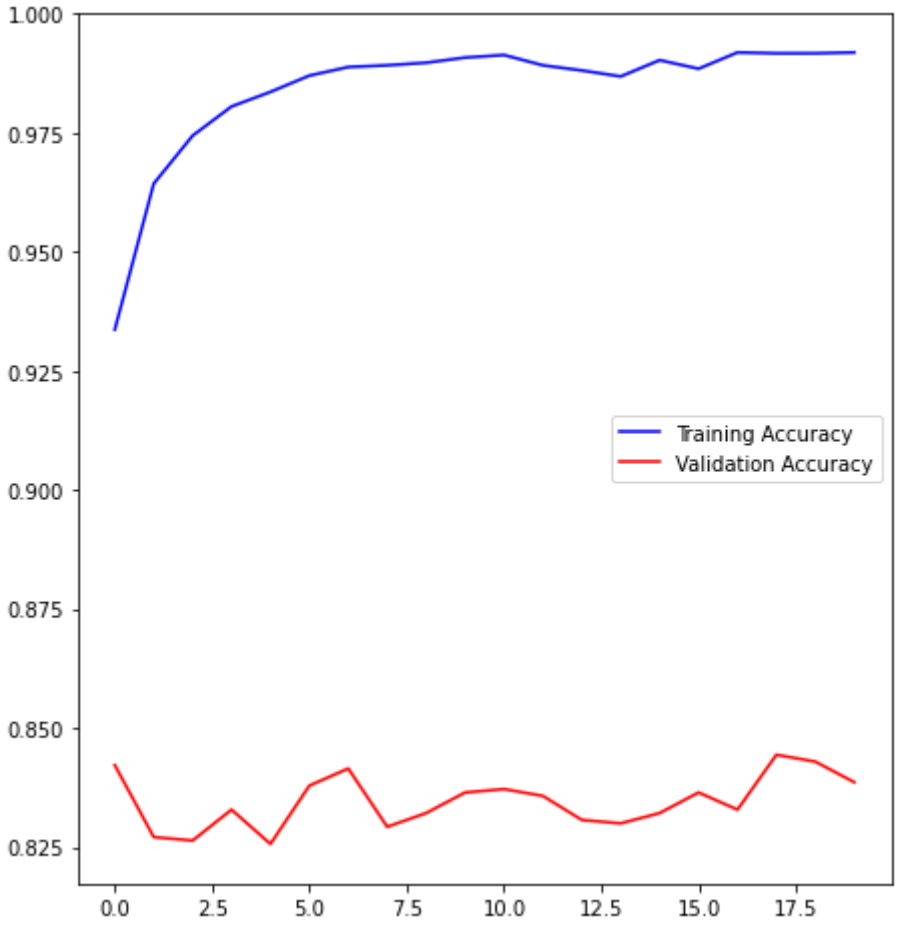


FIGURE 5.28 – *Graphe "Accuracy" de la dixième expérimentation selon les epochs*

5.5 DIFFÉRENTS RÉSULTATS

5.5.1 Résumé de toutes les expérimentations

Modéle	Paramétre	Optimizer	Batch-size	Epoch	Précision	Rappel	F-mesure	Acc	Meilleur Loss	Temps GPU(s)
Dataset1										
1	CNN(250,7) BiLSTM(250) Dense(250) Dropout(0.3)	RMSprop lr=0.001	32	15	0.79	0.81	0.80	0.81	0.3923	1193
2	BiLSTM(250) Dense(400) Dropout(0.3)	RMSprop lr=0.001	32	10	0.80	0.79	0.80	0.79	0.4004	775
2	BiLSTM(250) Dense(400) Dropout(0.3)	RMSprop lr=0.001	32	+5	0.80	0.79	0.79	0.79	3.23	395
Dataset2										
1+Nettoyage des données	CNN(250,7) BiLSTM(250) Dense(250) Dropout(0.3)	RMSprop lr=0.001	32	10	0.85	0.86	0.84	0.86	0.4034	226
1+Nettoyage des données	CNN(250,7) BiLSTM(250) Dense(250) Dropout(0.3)	Adam	64	10	0.85	0.86	0.85	0.86	0.6331	234
2+Nettoyage des données	BiLSTM(250) Dense(400) Dropout(0.3)	Adam lr=0.001	64	10	0.86	0.87	0.86	0.87	0.3493	93

Modéle	Paramétre	Optimizer	Batch-size	Epoch	Précision	Rappel	F-mesure	Acc	Meilleur Loss	Temps GPU(s)
1+Nettoyage des données	CNN(250,3) BiLSTM(250) Dense(250) Dropout(0.3)	Adam lr=0.01	32	15	0.60	0.78	0.68	0.78	0.5323	133
Dataset3										
2+Nettoyage des données +stopwords	BiLSTM(128) Dense(100) Dropout(0.2)	Adam lr=0.001	16	10	0.83	0.83	0.83	0.83	2.8426	81
1+Nettoyage des données +stopwords	CNN(128,3) BiLSTM(128) Dense(128) Dropout(0.3)	Adam	32	100	0.84	0.84	0.84	0.84	0.9452	1121
1+Nettoyage des données +stopwords	CNN(250,5) BiLSTM(250) Dense(250) Dropout(0.3)	Adam lr=0.001	32	20	0.84	0.84	0.84	0.84	0.5535	224

TABLE 5.1 – *Résumé des expérimentations*

Après avoir effectuer plusieurs expérimentations ,nous avons prouvé qu'un mauvais choix de nombre d'epochs peut influer négativement sur les performances.
Malgré qu'en DL nous n'avons pas besion beaucoup de prétraimtement ,mais le fait de les appliquer on peut réduire le nombre de paramètres donc on peut réduire le temps d'exécution et la différence est bien claire sur le tableau.

5.5.2 Comparaisons des résultats par rapport a l'état de l'art

Dataset1				
	Précision	***Rappel***	***F-mesure***	***Acuuracy***
Notre approche Model1	0.79	**0.81**	**0.8**	**0.81**
Notre approche Model2	0.80	0.79	**0.8**	0.79
Mubarak et al. [2017]	**0.98**	0.45	0.59	/
Dataset2				
	Précision	***Rappel***	***F-mesure***	***Acuuracy***
Notre approche Model1	0.85	0.86	0.85	0.86
Notre approche Model2	0.86	**0.87**	**0.86**	0.87 (EarlyStopping 0.8704)
Mohaouchane et al. [2019]	0.861	0.834	0.84	**0.878**
Dataset3				
	Précision	***Rappel***	***F-mesure***	***Acuuracy***
Notre approche Model1	**0.84**	**0.84**	**0.84**	**0.84**
Notre approche Model2	0.83	0.83	0.83	0.83
Mulki et al. [2019]	0.90	0.89	0.896	0.903
Mubarak et Darwish [2019]	0.67	0.65	0.64	/

TABLE 5.2 – *Comparaisons des résultats par rapport a l'état de l'art*

5.5.3 Interface Graphique

Afin de rendre accessible notre approche et faciliter/augmenter son utilisation, nous avons décidé de lui définir une interface graphique simple.
Nous avons utilisé Streamlit qui est une bibliothèque Python open source qui facilite la création et le partage des applications Web personnalisées pour l'apprentissage automatique et data science en générale. Dans cette interface :

— L'utilisateur peut avoir des information sur le modèle qui tourne au fond et peut aussi enregistrer ses informations en photo(5.30).

— L'utilisateur met en entrée le fichier(.xslsx) qui ne dépasse pas les 200 MB qui contient les textes qui veut les exploités (5.29).
— Une fois le fichier télechargé , il aura en sortie les informations sur les colonnes le nombre de ligne contenu dans ce fichier ainsi qu'un affichage du dataset (5.31), il peut afficher tout le dataset dans un nouvel onglet (5.32).
— Il peut aussi effectuer des prétraitements et les appliqués a son dataset5.33 (il a la possibilité de passer sans effectuer des prétraitement) .
— Après qu'il termine l'étape précédente, il peut choisir :

1. Train% :Pour définir combien il garde pour l'apprentissage ,la valeur par défaut est à 80, la minimale a 60 et maximale a 95.
2. Batch size :Ce champ prend une valeur par défaut a 32 la minimale a 16 , la maximale a 128(Pour ne pas alourdir le serveur).
3. Epochs : Prend au minimum 3 (pour ne pas avoir un underfitting) au maximum 100(Pour ne pas alourdir le serveur), valeur par défaut est a 55.34.

— Après lancement du "Begin" et après attente pour pouvoir terminé les calculs 5.35,les résultats d'exécution(du fit) 5.36 s'affichent , ainsi que la matrice de confusion,accuracy aussi et une table qui contient les différentes performances (5.37).
— Il a la possibilité aussi d'avoir le graphe d'accuracy et loss (pour training et validation),il peut aussi l'enregistrer5.38 .
— Au final il peut avoir la prédiction d'un texte écrit dans une zone de texte (La figures5.39 inclue deux exemples de prédiction).

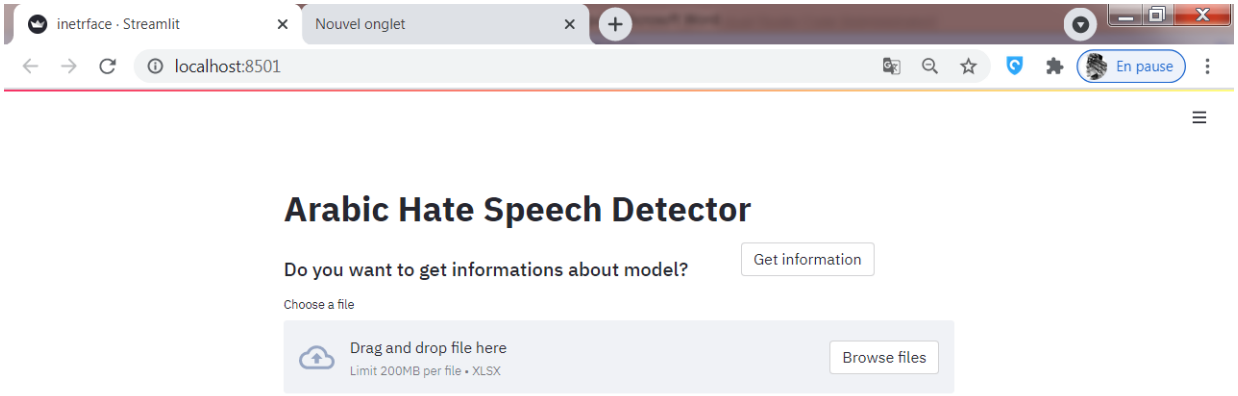


FIGURE 5.29 – *Interface Web (première page)*

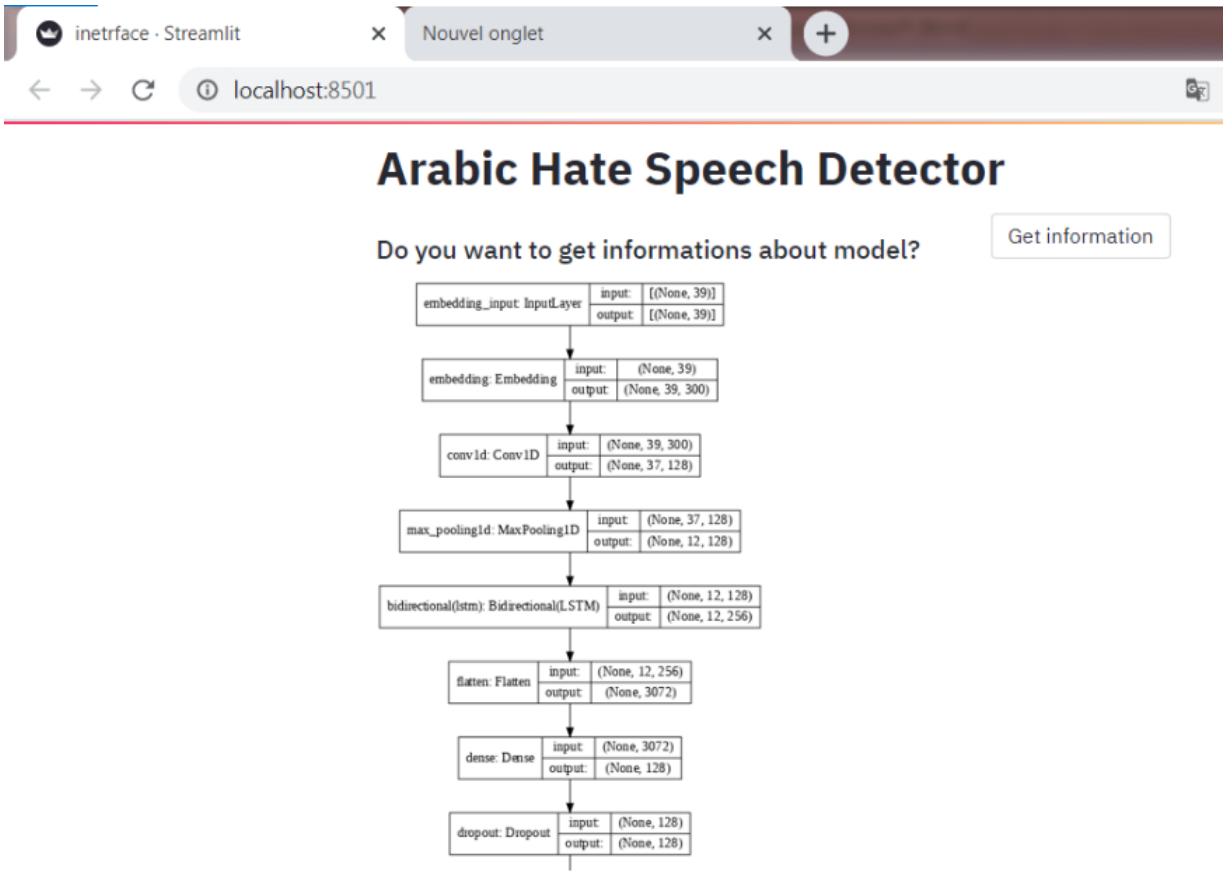


FIGURE 5.30 – *Affichage des informations du modèle*

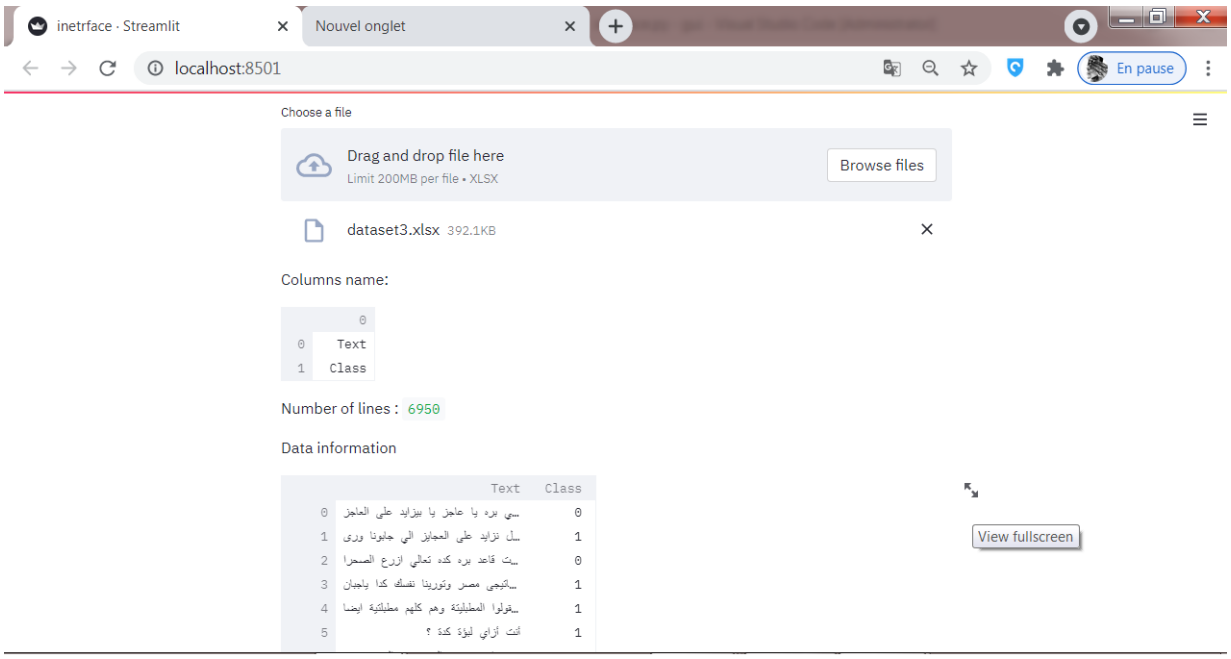


FIGURE 5.31 – *Affichage des informations sur le dataset*

Text	Class
" انت فعلا قذر عشان تتمت في بلدك بالطريقة دي .. ملعون ابوك يا شيخ"	1
شماتة ايه بس هو لما نشاور ع غباء المسؤلين يبقى شماتة	1
لا دي شماتة .. و عايز جنازة يتسع فيها لطم ميعرفش يخي قمية الحقد و الغل اللي عنده ابدا .. انسان قذر	1
"D-: :-D انا اعرف ان الحقد بيبقي علي حاجه كويسه D-: هو في حاجه اساسا تستاهل يبقي عليها حقد"	0
ايوة حقد و شماتة لان لما بيبقي فيه مناسبة حلوة في البلد بيتخرس زي سيده البرادعي	1
"انت ايه كم الفاظ الكراهيه اللي انت فيها ده انت عايزه ايه"	0
🙄 وحشتنا اوي ياباسم	0
😂😂😂 كابتشن كابتشن الحياة منغيرك قميئة و مزرية	0
الله يرحمه ابوك مخلف خول رسمي	1
وكمان عراقي فاشل روح ياشاطر الحب بعيد بلا بابا شوفلك واحد امريكاني يظبطك	1
اعتذر عن عدم الرد على التويتات. مشغول لشوشتي في فعاليات المجلس الأعلى للعالم. تحيوه كام ريختر؟	0
عايزة 6 ريختر و يكون معه تسونامي صغير و الامطار علي شكل كرانيش و يا ريت بينك	0
راح عليك انت شوية شغل ...لوز اللوز ...اسكريبت البرنامج كان هيتكتب لوحده وتمد ايدك وانت مغمض وتسحب منه ومايخلصش	0
نصف سوا علشان الدابت يا باسيييييييييييييم	0
هو إيه بالضبط؟	0
"بجد يا باسم كلنا مفتقدينك ربنا يرجعك بالسلامه"	0
خمسسكة يا باسم خمسة مواااه	0
.حابة أشاركم الفعاليات وقررت أمشي بحتية	0

FIGURE 5.32 – *Affichage du dataset en entier*

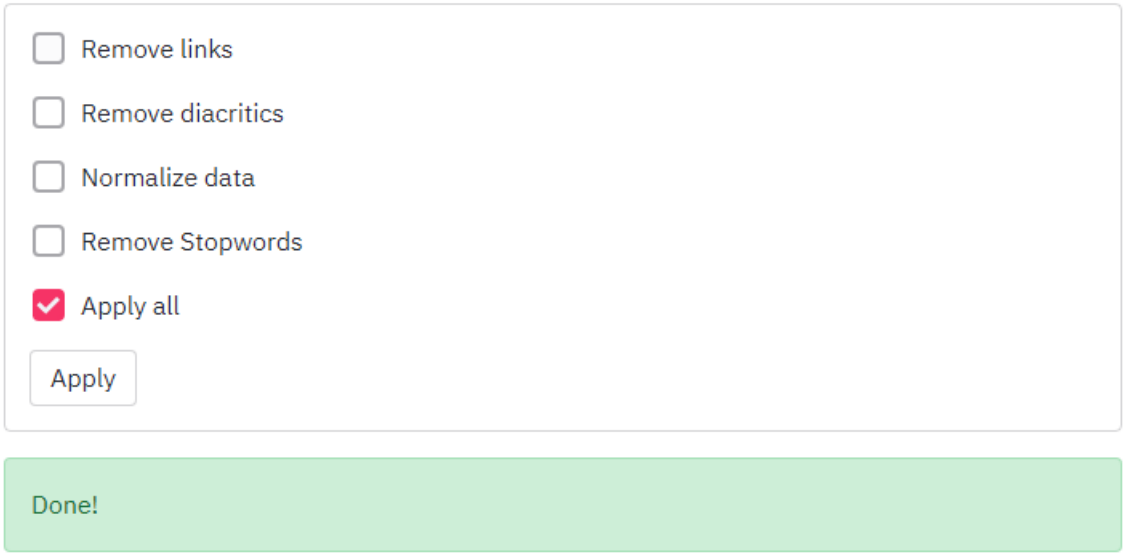


FIGURE 5.33 – *Application de prétraitement*

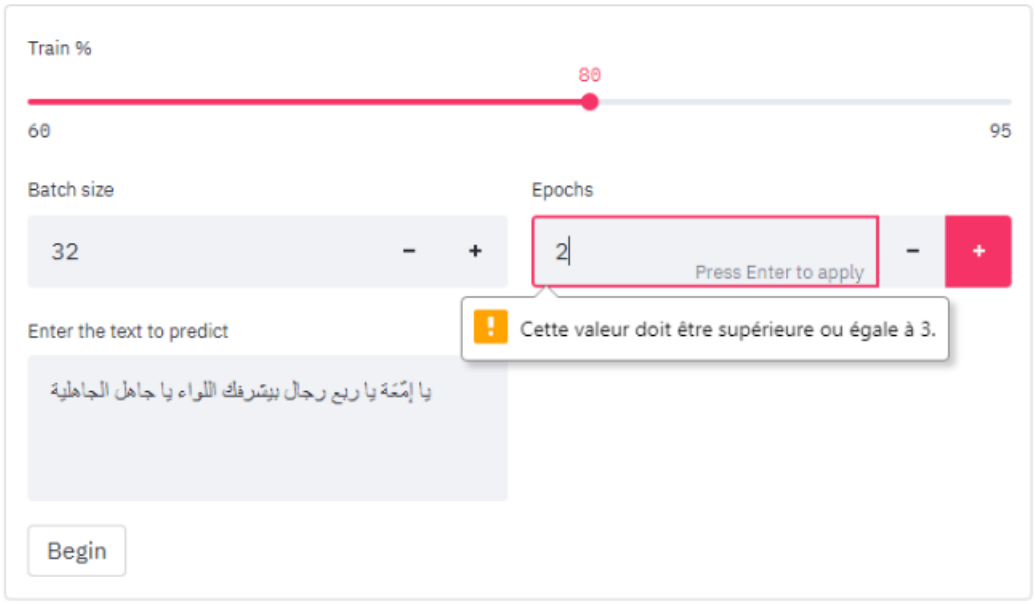


FIGURE 5.34 – *Paramètres du fit*

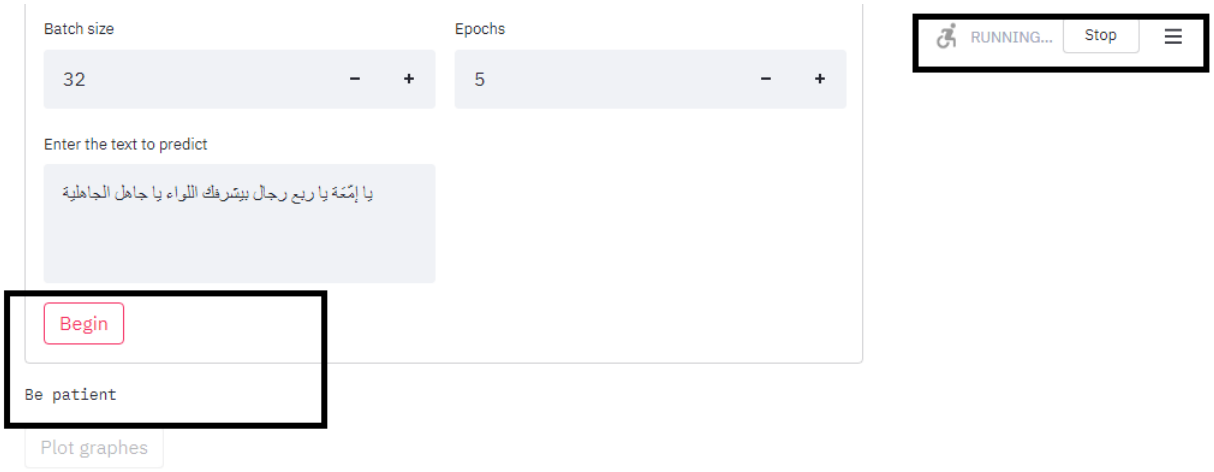


FIGURE 5.35 – *Lancement des calculs*

	epoch	accuracy	loss	val_accuracy	val_loss
0	0	0.6282	0.6709	0.6806	0.6202
1	1	0.7656	0.4917	0.7842	0.4612
2	2	0.8840	0.2830	0.7957	0.4528
3	3	0.9480	0.1395	0.8058	0.4802
4	4	0.9707	0.0831	0.8014	0.6995

FIGURE 5.36 – *Résultats d'exécution (du fit)*

Confusion Matrix

Confusion Matrix

	0	1
0	769	82
1	194	345

Accuracy

Accuracy_Score: 0.8

PERFORMANCE

	precision	recall	f1-score	support
0	0.7985	0.9036	0.8479	851
1	0.8080	0.6401	0.7143	539
accuracy	0.8014	0.8014	0.8014	0.8014
macro avg	0.8033	0.7719	0.7811	1390
weighted avg	0.8022	0.8014	0.7961	1390

FIGURE 5.37 – *Affichage des différentes performances.*

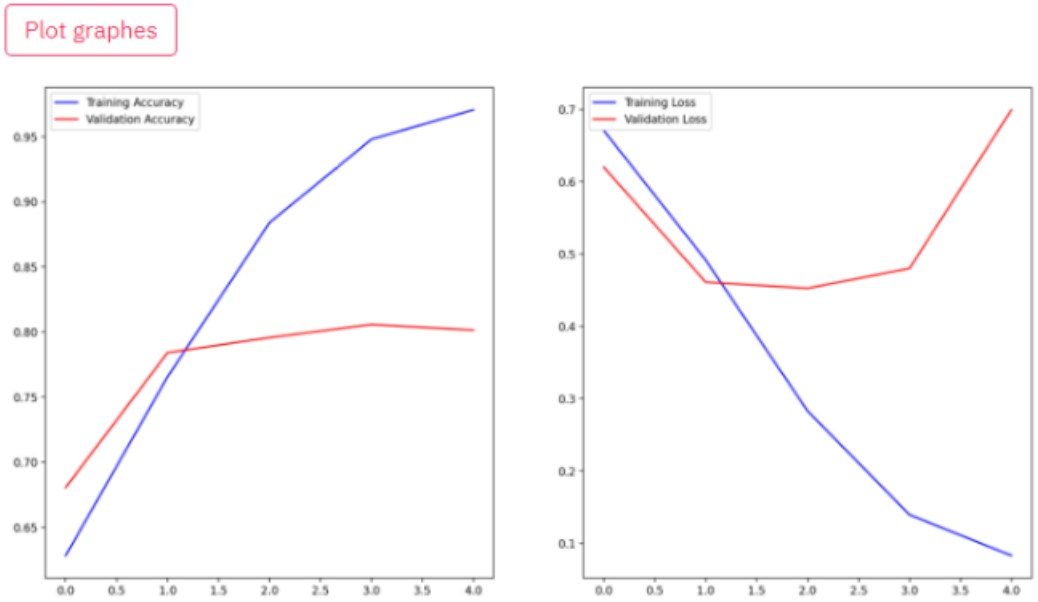


FIGURE 5.38 – *Affichage des graphes*

FIGURE 5.39 – *Prédiction de deux exemples différents*

Conclusion générale

Dans ce mémoire de master II, nous avons présenté deux méthodes efficaces pour classifier les différents commentaires qui existent sur les réseaux sociaux étant Hate ou bien non-Hate dans n'importe quel dialecte arabe tout en profitant des caractéristiques puissantes des réseaux de neurones en deep learning.

Nous avons défini deux modèles : une hybridation CNN-BiLSTM et un modèle BiLSTM seulement pour classifier le texte avec une représentation de mot en utilisant AraVec .Les différentes expérimentations que nous avons effectué avec les différents datasets et avec les différents dialectes , ont montré l'efficacité de nos modèles pour la détection du Hate Speech.

L'originalité de notre travail réside dans la prise en compte du multi dialecte dans l'analyse automatique des textes pour la détection du Hate Speech ou bien les discours haineux sur les différents réseaux sociaux .
Ce travail constitue ainsi une étape vers une élimination automatique du Hate Speech à partir de nos réseaux sociaux , de plus cela peut être aussi appliquer sur les autres pages / médias telles que les forums les messages visualisées aux télévisions etc ...
Espérant avoir des médias sans Hate Speech dans le futur prochain.

Perspectives :

Plusieurs perspectives à notre travail sont identifiées. Nous les présentons dans les points suivants :

- Faire plus d'expérimentations en changeant plus de paramètres : nombre de neurones de la couche entièrement connectée, le taux de drop out, etc.
- Modifier l'architecture en ajoutant ou en supprimant une/des couche(s).
- Essayer d'autre représentation des mots en utilisant fastText ,GloVe ou autre .
- Améliorer le graphisme et enrichir notre site avec d'autres fonctionnalités .

Bibliographie

Ehab Abozinadah, Alex Mbaziira, et James Jones. Detection of abusive accounts with arabic tweets. *Int. J. Knowl. Eng.-IACSIT*, 1(2) :113–119, 2015.

Ahmad Al Sallab, Hazem Hajj, Gilbert Badaro, Ramy Baly, Wassim El-Hajj, et Khaled Shaban. Deep learning models for sentiment analysis in arabic. Dans *Proceedings of the second workshop on Arabic natural language processing*, pages 9–17, 2015.

Azalden Alakrot. *Detection of anti-social behaviour in online communication in Arabic*. PhD thesis, University of Limerick, May 2019.

Azalden Alakrot, Liam Murray, et Nikola Nikolov. Dataset construction for the detection of anti-social behaviour in online communication in arabic. *Procedia Computer Science*, 142 :174–181, 2018.

Nuha Albadi, Maram Kurdi, et Shivakant Mishra. Are they our brothers ? analysis and detection of religious hate speech in the arabic twittersphere. Dans *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 69–76. IEEE, 2018.

Raghad Alshalan et Hend Al-Khalifa. A deep learning approach for automatic hate speech detection in the saudi twittersphere. *Applied Sciences*, 10 (23) :8614, 2020.

Ramy Baly, Gilbert Badaro, Georges El-Khoury, Rawan Moukalled, Rita Aoun, Hazem Hajj, Wassim El-Hajj, Nizar Habash, et Khaled Shaban. A characterization study of arabic twitter data with a benchmarking for state-of-the-art opinion mining models. Dans *Proceedings of the third Arabic natural language processing workshop*, pages 110–118, 2017.

Arijit Ghosh Chowdhury, Aniket Didolkar, Ramit Sawhney, et Rajiv Shah. Arhnet-leveraging community interaction for detection of religious hate speech in arabic. Dans *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics : Student Research Workshop*, pages 273–280, 2019.

Ibrahim Abu Farha et Walid Magdy. Mazajak : An online arabic sentiment analyser. Dans *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 192–198, 2019.

Hossam Faris, Ibrahim Aljarah, Maria Habib, et Pedro A Castillo. Hate speech detection using word embedding and deep learning in the arabic language context. Dans *ICPRAM*, pages 453–460, 2020.

Teona Gelashvili. Hate speech on social media : Implications of private regulation and governance gaps, 2018.

Imane Guellil, Ahsan Adeel, Faical Azouaou, Sara Chennoufi, Hanene Maafi, et Thinhinane Hamitouche. Detecting hate speech against politicians in arabic community on social media. *International Journal of Web Information Systems*, 2020.

Maha Heikal, Marwan Torki, et Nagwa El-Makky. Sentiment analysis of arabic tweets using deep learning. *Procedia Computer Science*, 142 :114–122, 2018. ISSN 1877-0509. URL https://www.sciencedirect.com/science/article/pii/S1877050918321689. Arabic Computational Linguistics.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02) :107–116, 1998.

Rita Izsak-Ndiaye. Report of the special rapporteur on minority issues, rita izsak. page 22 p., Jan 2015. URL http://digitallibrary.un.org/record/793592. Includes a thematic discussion on hate speech and incitement to hatred against minorities in the media.

Diederik Kingma et Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

Philip Klostermann. Youtube comment scraper. web-based application, 2015. URL http://ytcomments.klostermann.ca/. visité le :2021-05-2.

Leena Lulu et Ashraf Elnagar. Automatic arabic dialect classification using deep learning models. *Procedia computer science*, 142 :262–269, 2018.

Murat Mengü et Seda Mengü. Violence and social media. *Athens Journal of Mass Media and Communications*, 1(3) :211–227, 2015.

Tomas Mikolov, Kai Chen, Greg Corrado, et Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

Ammar Mohammed et Rania Kora. Deep learning approaches for arabic sentiment analysis. *Social Network Analysis and Mining*, 9(1) :1–12, 2019.

Hanane Mohaouchane, Asmaa Mourhir, et Nikola Nikolov. Detecting offensive language on arabic social media using deep learning. Dans *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 466–471, 2019.

Hamdy Mubarak et Kareem Darwish. Arabic offensive language classification on twitter. Dans *International Conference on Social Informatics*, pages 269–276. Springer, 2019.

Hamdy Mubarak, Kareem Darwish, et Walid Magdy. Abusive language detection on Arabic social media. Dans *Proceedings of the First Workshop on Abusive Language Online*, pages 52–56, Vancouver, BC, Canada, Août 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W17-3008.

Hala Mulki, Hatem Haddad, Chedi Bechikh Ali, et Halima Alshabani. L-HSAB : A Levantine Twitter dataset for hate speech and abusive language. Dans *Proceedings of the Third Workshop on Abusive Language Online*, pages 111–118, Florence, Italy, Août 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W19-3512.

Eslam Omara, Mervat Mosa, et Nabil Ismail. Deep convolutional network for arabic sentiment analysis. Dans *2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC)*, pages 155–159. IEEE, 2018.

Fadi Salem. The arab social media report 2017 : Social media and the internet of things : Towards data-driven policymaking in the arab world. *Dubai : MBR School of Government*, 7, 2017.

Abu Bakr Soliman, Kareem Eissa, et Samhaa El-Beltagy. Aravec : A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117 :256–265, 2017.

Vedran Vukotić, Vincent Claveau, et Christian Raymond. IRISA at DeFT 2015 : Supervised and Unsupervised Methods in Sentiment Analysis. Dans *DeFT, Défi Fouille de Texte, joint à la conférence TALN 2015*, Actes de l'atelier DeFT, Défi Fouille de Texte, joint à la conférence TALN 2015, Caen, France, Juin 2015. URL https://hal.archives-ouvertes.fr/hal-01226528.

Scanner ce code QR pour avoir le code source et la version pdf de ce document .