Une approche neuronale pour l'analyse d'opinions en arabe

Amira Barhoumi

جـــامعة صفـــاقس
University of Sfax
Université de Sfax

Le Mans Université

THESE DE DOCTORAT DE

LE MANS UNIVERSITE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Amira BARHOUMI

Une approche neuronale pour l'analyse d'opinions en arabe

Thèse présentée et soutenue à Le Mans, le 23/11/2020
Unité de recherche : LIUM
Thèse N° : 2020LEMA1022

Rapporteurs avant soutenance :

Kamel Smaïli	Professeur, Université de Lorraine, France
Nadia Essoussi	Professeur, Université de Tunis, Tunisie

Composition du Jury :

Président :	Anthony Larcher	Professeur, Le Mans Université, France
Examinateurs :	Emmanuel Morin	Professeur, Université de Nantes, France
Dir. de thèse :	Yannick Estève	Professeur, Université d'Avignon, France
	Lamia Hadrich Belguith	Professeur, FSEGS, Université de Sfax, Tunisie
Co-encadrant de thèse :	Nathalie Camelin	Maître de conférences, Le Mans Université, France

Invité
Chafik Aloulou — Maître de conférences, FSEGS, Université de Sfax, Tunisie

جـــامعة صفـــاقس
University of Sfax
Université de Sfax

Le Mans Université

Titre : Une approche neuronale pour l'analyse d'opinions en arabe

Mots clés : Analyse d'opinions, Réseaux de neurones convolutifs, Réseaux de neurones récurrents, représentation vectorielle continue, langue arabe.

Résumé : Cette thèse s'inscrit dans le cadre de l'analyse d'opinions en arabe. Son objectif consiste à déterminer la polarité globale d'un énoncé textuel donné écrit en Arabe standard moderne (ASM) ou dialectes arabes. Cette thématique est un domaine de recherche en plein essor et a fait l'objet de nombreuses études avec une majorité de travaux actuels traitant des langues indo-européennes, en particulier la langue anglaise. Une des difficultés à laquelle se confronte cette thèse est le traitement de la langue arabe qui est une langue morphologiquement riche avec une grande variabilité des formes de surface observables dans les données d'apprentissage. Nous souhaitons pallier ce problème en produisant, de manière totalement automatique et contrôlée, de nouvelles représentations vectorielles continues (en anglais *embeddings*) spécifiques à la langue arabe.

Notre étude se concentre sur l'utilisation d'une approche neuronale pour améliorer la détection de polarité, en exploitant la puissance des embeddings. En effet, ceux-ci se sont révélés un atout fondamental dans différentes tâches de traitement automatique des langues naturelles (TALN).

Notre contribution dans le cadre de cette thèse porte plusieurs axes. Nous commençons, d'abord, par une étude préliminaire des différentes ressources d'embeddings de mots pré-entraînés existants en langue arabe.

Ces embeddings considèrent les mots comme étant des *unités séparées par des espaces* afin de capturer, dans l'espace de projection, des similarités sémantiques et syntaxiques. Ensuite, nous nous focalisons sur les spécificités de la langue arabe en proposant des embeddings spécifiques pour cette langue. Les phénomènes comme l'agglutination et la richesse morphologique de l'arabe sont alors pris en compte. Ces embeddings spécifiques ont été utilisés, seuls et combinés, comme entrée à deux réseaux neuronaux (l'un convolutif et l'autre récurrent) apportant une amélioration des performances dans la détection de polarité sur un corpus de revues.

Nous proposons une analyse poussée des embeddings proposées. Dans une évaluation intrinsèque, nous proposons un nouveau protocole introduisant la notion de la stabilité de polarités (*sentiment stability*) dans l'espace d'embeddings. Puis, nous proposons une analyse qualitative extrinsèque de nos embeddings en utilisant des méthodes de projection et de visualisation.

جـــامعة صفـــاقس
University of Sfax
Université de Sfax

Le Mans Université

Title : Neural approach for Arabic sentiment analysis

Keywords: Sentiment analysis, Convolutional neural network, Recurrent neural Network, embeddings, arabic language

Abstract : My thesis is part of Arabic sentiment analysis. Its aim is to determine the global polarity of a given textual statement written in MSA or dialectal arabic. This research area has been subject of numerous studies dealing with Indo-European languages, in particular English. One of difficulties confronting this thesis is the processing of Arabic. In fact, Arabic is a morphologically rich language which implies a greater sparsity : we want to overcome this problem by producing, in a completely automatic way, new arabic specific embeddings.

Our study focuses on the use of a neural approach to improve polarity detection, using embeddings. These embeddings have revealed fundamental in various natural languages processing tasks (NLP).

Our contribution in this thesis concerns several axis. First, we begin with a preliminary study of the various existing pre-trained word embeddings resources in arabic. These embeddings consider words as space separated units in order to capture semantic and syntactic similarities in the embedding space.

Second, we focus on the specifity of Arabic language. We propose arabic specific embeddings that take into account agglutination and morphological richness of Arabic. These specific embeddings have been used, alone and in combined way, as input to neural networks providing an improvement in terms of classification performance. Finally, we evaluate embeddings with intrinsic and extrinsic methods specific to sentiment analysis task. For intrinsic embeddings evaluation, we propose a new protocol introducing the notion of *sentiment stability* in the embeddings space. We propose also a qualitaive extrinsic analysis of our embeddings by using visualisation methods.

Remerciements

Cette thèse est le fruit d'échanges, de conseils et de soutiens d'un grand nombre de personnes auxquelles je tiens à adresser mes remerciements et exprimer ma reconnaissance.

Je tiens tout d'abord à exprimer mes vifs remerciements à mes encadrants de thèse pour leurs disponibilités, leurs soutiens, leurs conseils ciblés et précis, leurs encouragements ainsi que pour leur patience.

Je remercie mon directeur de thèse en France, M. Yannick Estève, pour ses précieux conseils, ses encouragements, sa bonne humeur et son soutien moral. Malgré ses responsabilités, il a su être présent quand il le fallait. Je remercie également, ma directrice de thèse en Tunisie, Lamia Hadrich Belguith, pour son encouragement, ses conseils et critiques constructives.

J'exprime mes plus profonds remerciements à ma co-encadrante en France, Mme Nathalie Camelin, pour sa confiance, pour le temps qu'elle m'a accordé et pour la qualité de son encadrement. Je la remercie également pour ses qualités personnelles et humaines qui ont beaucoup contribué à la réalisation de ce travail. Par ailleurs, je remercie M. Chafik Aloulou, mon co-encadrant en Tunisie, pour ses conseils utiles et ses recommandations précieuses.

Ce que j'ai appris en travaillant avec mes encadrants ne se limite pas à l'aspect scientifique mais s'étend aux aspects humain et relationnel. Je les remercie infiniment.

Je tiens aussi à exprimer mes remerciements à Mme Nadia Essoussi, Professeur à l'Université de Tunis, ainsi que M. Kamel Smaïli, Professeur à l'Université de Lorraine, qui ont accepté de juger ce travail et d'en être les rapporteurs.

Je tiens aussi à remercier les examinateurs de ma thèse M. Emmanuel Morin, professeur à l'univrsité de Nantes, et M. Anthony Larcher, professeur à le Mans université, d'avoir accepté de juger cette thèse et pour l'intérêt qu'ils ont porté à mon travail.

Je remercie tous les membres du Laboratoire LIUM pour leur accueil chaleureux et leur convivialité. Je tiens également à souligner le formidable cadre de travail que le LIUM offre aux doctorants. C'est réellement une chance de pouvoir réaliser une thèse dans un environnement de cette qualité. Je remercie également les membres du laboratoire MIRACL.

Rien n'aurait été possible sans le soutien de ma famille. Merci encore à vous, rien n'a su me motiver davantage que votre appui et la confiance que vous m'avez toujours accordée.

Un grand merci à ma sœur Omayma qui n'a pas cessé de m'encourager et de m'offrir des conditions favorables durant ces années d'études. Je remercie également ma sœur Oumaya et mon frére Amir.

Je dédie ce travail à mes parents. Merci papa pour tes recommandations qui m'ont incité à persister malgré les difficultés, persévérer face aux obstacles et progresser dans ma vie professionnelle. Merci maman pour ta tendresse et ton amour et ses encouragements durant la période de ma thèse.

Enfin, à tous ceux que je n'ai pas pu citer, auxquels je réitère mes sincères remerciements.

À vous tous, Merci !

Table des matières

Introduction **1**

1 Analyse d'opinions **5**
1.1 Notion d'analyse d'opinions . . . 6
1.1.1 Étymologie . . . 6
1.1.2 Modélisation formelle d'opinions . . . 6
1.1.3 Types d'opinions . . . 8
1.1.4 Niveaux d'analyse d'opinions . . . 9
1.1.4.1 Niveau document . . . 9
1.1.4.2 Niveau phrase . . . 10
1.1.4.3 Niveau aspect . . . 10
1.1.4.4 Niveau mot . . . 10
1.2 Approches d'analyse d'opinions . . . 10
1.2.1 Approche symbolique . . . 11
1.2.2 Approche numérique . . . 14
1.2.2.1 Apprentissage automatique . . . 15
1.2.2.2 Apprentissage profond . . . 16
1.2.3 Approche hybride . . . 17
1.3 Challenges de l'analyse d'opinions en arabe . . . 18
1.3.1 Manque de ressources . . . 18
1.3.2 Opinion spam . . . 18
1.3.3 Ironie et sarcasme . . . 18
1.4 Domaines d'application de l'analyse d'opinions . . . 19
1.5 Conclusion . . . 20

2 Langue arabe et analyse d'opinions **21**
2.1 Présentation de la langue arabe . . . 22

2.1.1 Caractéristiques de l'arabe standard moderne . . . 23
2.1.2 Caractéristiques de l'arabe dialectal . . . 27
2.1.2.1 Origine . . . 27
2.1.2.2 Normalisation d'écriture . . . 28
2.1.2.3 Traitement automatique de l'AD . . . 28
2.1.2.4 Différences entre AD et ASM . . . 29
2.1.3 Systèmes d'écriture pour l'arabe . . . 30
2.1.3.1 Système à base de caractères arabes . . . 31
2.1.3.2 Système à base de caractères latins : Arabizi . . . 32
2.2 Prétraitement automatique de la langue arabe . . . 32
2.2.1 Segmentation . . . 33
2.2.2 Lemmatisation . . . 33
2.2.3 Stemming . . . 34
2.2.4 Light stemming . . . 34
2.3 Méthodes de construction de ressources . . . 34
2.3.1 Lexiques polarisés . . . 34
2.3.2 Corpus . . . 38
2.3.3 Résumé des ressources existantes . . . 40
2.3.3.1 Corpus . . . 40
2.3.3.2 Lexiques polarisés . . . 41
2.4 État de l'art de l'analyse d'opinions en arabe . . . 42
2.4.1 Approche symbolique . . . 43
2.4.2 Approche numérique . . . 43
2.4.3 Approche hybride . . . 44
2.5 Spécificité de la langue arabe pour l'analyse d'opinions . . . 45
2.5.1 Catégories d'arabe . . . 45
2.5.2 Dialecte arabe . . . 46
2.5.3 Polysémie . . . 46
2.5.4 Non voyellation . . . 47
2.5.5 Agglutination . . . 47
2.5.6 Entitées nommées . . . 48
2.5.7 Négation . . . 48
2.5.8 Système d'écriture . . . 48
2.6 Conclusion . . . 48

3 Réseaux de neurones pour l'analyse d'opinions en arabe **51**
3.1 Réseaux de neurones : généralités . . . 52
3.1.1 Définition . . . 52
3.1.1.1 Neurone formel . . . 52

3.1.1.2 Perceptron multi-couches . . . 53
3.1.2 Apprentissage . . . 54
3.1.3 Résumé . . . 57
3.2 Réseaux de neurones convolutifs . . . 58
3.2.1 Entrée du CNN . . . 59
3.2.2 Convolution . . . 60
3.2.3 Pooling . . . 62
3.2.4 Couches entièrement connectées . . . 63
3.2.5 Sortie du CNN . . . 63
3.3 Réseaux de neurones récurrents . . . 64
3.3.1 Présentation des RNN . . . 64
3.3.2 Long-Short Term Memory . . . 68
3.3.3 Long-Short Term Memory Bidirectionnels . . . 69
3.4 Représentations continues de mots . . . 69
3.4.1 Méthodes de construction . . . 70
3.4.1.1 Word2vec . . . 71
3.4.1.2 FastText . . . 73
3.4.1.3 Autres Méthodes : Elmo et Bert . . . 73
3.4.2 Techniques d'évaluation . . . 74
3.4.2.1 Méthodes d'évaluation intrinsèque . . . 74
3.4.2.2 Méthodes d'évaluation extrinsèque . . . 74
3.5 Réseaux de neurones pour l'AO en arabe : état de l'art . . . 75
3.5.1 Méthodes neuronales . . . 75
3.5.2 Prétraitements . . . 78
3.5.3 Synthèse . . . 79
3.6 Conclusion . . . 80

4 Systèmes neuronaux pour l'analyse d'opinions en arabe : une étude préliminaire 81
4.1 Systèmes d'analyse d'opinions . . . 82
4.1.1 Architecture neuronale . . . 82
4.1.2 Embeddings de mots utilisés . . . 83
4.2 Cadre expérimental . . . 85
4.2.1 Corpus . . . 85
4.2.1.1 Composition du corpus . . . 85
4.2.1.2 Préparation des ensembles Train, Dev et Test . . . 86
4.2.1.3 Pré-traitement du corpus . . . 87
4.2.2 Lexique polarisé . . . 87
4.2.3 Métriques d'évaluation . . . 89
4.3 Premières expériences . . . 89

4.3.1 Impact des pré-traitements du corpus . . . 90
4.3.2 Premières évaluations du CNN . . . 91
4.3.2.1 Impact du choix d'embeddings d'entrée sur le CNN . . . 91
4.3.2.2 Premières analyses . . . 93
4.3.3 Représentation des documents . . . 94
4.3.3.1 Longueur du document . . . 95
4.3.3.2 Type de padding/Truncating . . . 95
4.3.3.3 Impact sur les performances . . . 96
4.4 Embeddings de mots : étude qualitative . . . 97
4.4.1 Couverture . . . 97
4.4.2 Étude du voisinage . . . 99
4.5 Conclusion . . . 100

5 Embeddings spécifiques à la langue arabe 103
5.1 Motivations . . . 104
5.2 Analyse extrinsèque des embeddings spécifiques à l'arabe . . . 105
5.2.1 Évaluation des embeddings simples . . . 105
5.2.1.1 Protocole . . . 106
5.2.1.1.a Construction des embeddings spécifiques à l'arabe . . . 106
5.2.1.1.b Architectures neuronales . . . 110
5.2.1.1.c Pourcentage de l'ASM dans LABR . . . 112
5.2.1.2 Résultats . . . 113
5.2.1.3 Comparaison avec les embeddings de mots pré-entrainés existants 114
5.2.1.3.a Couverture . . . 114
5.2.1.3.b Étude de voisinage . . . 116
5.2.2 Évaluation de combinaison des sorties de systèmes . . . 118
5.2.2.1 Protocole . . . 118
5.2.2.2 Résultats . . . 118
5.2.2.2.a Analyse du non consensus . . . 119
5.2.2.2.b Analyse du consensus . . . 120
5.2.2.2.c Fiabilité des données d'apprentissage . . . 120
5.2.3 Évaluation de combinaison des entrées de systèmes . . . 121
5.2.3.1 Protocole . . . 121
5.2.3.2 Résultats . . . 122
5.3 Analyse qualitative des embeddings spécifiques à l'arabe . . . 123
5.3.1 Méthodologie . . . 124
5.3.1.1 Méthode d'évaluation intrinsèque . . . 124
5.3.1.2 Méthode d'évaluation extrinsèque . . . 125
5.3.1.3 Espaces d'embeddings . . . 125

5.3.2 Résultats . . . 126
5.3.2.1 Résultats de l'évaluation intrinsèque . . . 126
5.3.2.2 Résultats de l'évaluation extrinsèque . . . 128
5.3.3 Discussion . . . 129
5.4 Conclusion . . . 131

6 Conclusion et perspectives **133**
6.1 Conclusion . . . 133
6.2 Perspectives . . . 135

A Embeddings spécifiques à la langue arabe : statistiques et performances **137**
A.1 Prototocole de représentation de documents : impact sur les performances . . . 137
A.2 Résulats des Embeddings combinés . . . 137

B La traduction automatique pour l'AOA : une étude empirique **139**
B.1 Motivations . . . 140
B.2 État de l'art . . . 141
B.3 Méthodologie . . . 142
B.3.1 Systèmes d'analyse d'opinions . . . 144
B.3.2 Embeddings de documents . . . 144
B.4 Résultats et discussion . . . 145
B.4.1 Performances sur le corpus *LABR_ar* . . . 145
B.4.2 Performance sur le corpus *LABR_en* . . . 147
B.4.3 Discussion . . . 148
B.5 Conclusion . . . 150

Bibliographie personnelle **151**

Table des figures

1.1 Les approches en analyse d'opinions. 11

2.1 Structure générale d'un mot graphique en arabe [Saâdane, 2015]. 23
2.2 Hypothèse de la pyramide d'ambiguïté [Attia & Somers, 2008]. 24
2.3 Méthodes de construction de lexiques polarisés. 35

3.1 Analogie entre un neurone formel et un neurone biologique [Ghannay, 2017]. 53
3.2 Schéma d'un perceptron multi-couches (avec une seule couche cachée). 54
3.3 Application du dropout dans un réseau de neurones [Srivastava *et al.*, 2014]. 57
3.4 Réseau de neurones convolutif composé de deux couches de convolution et de pooling (*subsampling*), suivi de deux couches cachées et d'une couche de sortie [LeCun *et al.*, 1990]. 58
3.5 Mécanisme de construction de la matrice $M \in \mathbb{R}^{n \times k}$ correspondant à un document en se basant sur la table d'embeddings Embed. 60
3.6 Exemple d'application d'une convolution 1D sur une matrice d'embeddings M. . . . 61
3.7 Fonctionnement d'une opération de *Max-pooling* appliquée sur une carte de caractéristiques. 63
3.8 Représentation compacte d'un RNN. La flèche pointillée reflète la récurrence sur tous les éléments de la séquence d'entrée. 64
3.9 Schémas des variantes de RNN : (a) RNN de type Elman, (b) RNN de type Jordan et (c) RNN de type Elman et Jordan combiné. 65
3.10 Schéma d'un RNN avant. 66
3.11 Schéma d'un RNN arrière. 66
3.12 Illustration des unités récurrentes de type LSTM et GRU. Dans LSTM (a) : c et $\widetilde{c}$ sont respectivement la mémoire et le nouveau contenu de la mémoire. Dans GRU (b) : h et $\widetilde{h}$ sont respectivement l'activation et l'activation candidate [Cho *et al.*, 2014]. . . . 67
3.13 Schéma d'un BiLSTM. 69

3.14 Visualisation des plongements de mots dans un espace à deux dimensions [Turian *et al.*, 2010]. 70
3.15 Architectures de Skip-gram (A) et CBOW (B). 71
3.16 Exemples d'analogie sémantique et syntaxique [Mikolov *et al.*, 2013a]. 75

4.1 Architecture CNN de [Kim, 2014a] 82

5.1 Statistiques sur le corpus Global au niveau des différentes unités lexicales envisagées (mot, token, token\clitiques, lemme, light stemme et stemme) : taille du vocabulaire, taille du corpus et nombre d'unités apparaissant une fois. 108
5.2 Processus d'extraction d'embedding d'unité lexicale à base de caractères via un CNN. 110
5.3 Architecture du BiLSTM pour l'analyse d'opinions. 111
5.4 Graphiques des ratios de positivité et de négativité des mots polarisés dans l'espace d'embeddings de *ArEmb* entrainés avec word2vec. 117
5.5 *Sentiment stability* des embeddings de mots et de lemmes en fonction du type de corpus. 127

B.1 Méthodologie pour mesurer l'impact de la traduction automatique sur l'AOA. 143
B.2 Les versions DM et DBOW de l'algorithme *paragraph vector* [Le & Mikolov, 2014] 144

Liste des tableaux

1.1 Exemples de mots polarisés classés selon leurs catégories syntaxiques dans les langues anglais (En), français (Fr) et arabe (Ar). 12
1.2 Exemples de mots universels polarisés classés selon leurs catégories syntaxiques. . . 13
1.3 Exemples de descripteurs utilisés dans les techniques d'apprentissage automatique. . 16

2.1 Liste de clitiques (proclitiques et enclitiques) en arabe. 25
2.2 Agencements possibles des proclitiques. 26
2.3 Origine et sens de quelques mots empruntés utilisés dans l'AD. 27
2.4 Les signes diacritiques en arabe. 31
2.5 Exemples de caractères arabes et leurs équivalents en Arabizi. 32
2.6 Segmentation de quelques mots arabes. 33
2.7 Exemples montrant la réduction du vocabulaire de mots arabes avec la lemmatisation. 34
2.8 Correspondance entre le nombre d'étoiles et la polarité. 40
2.9 Corpus disponibles gratuitement pour l'analyse d'opinions en arabe. 41
2.10 Lexiques disponibles gratuitement utilisés dans cette thèse. 42
2.11 Liste de features utilisées pour AOA. 43
2.12 Différentes voyellisations possibles du mot جمل et leurs polarités correspondantes. . 47

3.1 Travaux réalisés en AOA. 76

4.1 Valeurs de quelques hyper-paramètres du CNN. 83
4.2 Taille de vocabulaire des différents ensembles d'embeddings pré-entrainés word2vec. 84
4.3 Taille de vocabulaire (nombre de mots) des différents ensembles d'embeddings pré-entrainés avec FastText. 85
4.4 Distribution du corpus LABR sur l'échelle de 5 étoiles 85
4.5 Répartition de LABR dans le cadre de la classification binaire 87
4.6 Répartition de LABR dans le cadre de la classification ternaire 87
4.7 Répartition de LABR dans le cadre de la classification quinaire 87

4.8 Exemples de mots polarisés dans *ArSentLex*. 88
4.9 Opérations de nettoyage du corpus par catégorie. 90
4.10 Exactitude (en %) du CNN sur LABR nettoyé par catégorie. 90
4.11 Performances du CNN sur le corpus de validation Dev avec les différents embeddings. 91
4.12 Performances du CNN sur le corpus de test avec les différents embeddings. 91
4.13 Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.arz* dans le cadre de la classification binaire. 92
4.14 Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.ar* dans le cadre de la classification ternaire. 92
4.15 Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.arz* dans le cadre de la classification quinaire. 92
4.16 Exemples de documents dans le corpus LABR. 94
4.17 Répartition (en %) des mots polarisés de *LABR_lex* dans l'ensemble Train du LABR. 94
4.18 Couverture de l'intégralité des documents de LABR avec le seuil 300 95
4.19 Informativité des différents segments du corpus Train et Dev de LABR 96
4.20 Performances du CNN sur le corpus de Test avec longueur 300. 97
4.21 Couverture (en %) du corpus LABR par les d'embeddings de type word2vec et FastText les plus performants. 98
4.22 Couverture (en %) du vocabulaire de LABR par les d'embeddings de type word2vec et FastText les plus performants. 98
4.23 Ratios de *positivité* (respectivement *négativité*) des mots positifs (respectivement négatifs) dont l'embedding existe à la fois dans *LABR_lex* et le corpus d'*embeddings* considéré. 100

5.1 Exemples de mots partageant la même forme fléchie. 104
5.2 Exemples montrant la réduction en vocabulaire entre mots et lemmes. 105
5.3 Statistiques sur le corpus *Global* avant et après nettoyage. 107
5.4 Taille des espaces d'embeddings spécifiques proposés pour l'arabe. 109
5.5 Évaluation de CNN avec les différents embeddings simples. 114
5.6 Rappel des meilleures performances du CNN sur le corpus de Test avec les embeddings de mots pré-entraînés existants (de type word2vec et FastText). 115
5.7 Rappel des couvertures du corpus LABR par les embeddings de mots pré-entraînés existants : wiki-cbow et *cc.arz*. 115
5.8 Rappel des ratios de positivité et de négativité des embeddings de mots pré-entraînés existants : wiki-cbow et *cc.arz*. 115
5.9 Biais de l'utilisation de LABR. 116
5.10 Ratios (en %) de positivité (respectivement de négativité) des mots positifs (respectivement négatifs) dont l'embedding existe à la fois dans *LABR_lex* et l'ensemble d'embeddings de mots d'*ArEmb*. 116

5.11 Évaluation des différents protocoles de combinaison des sorties de systèmes. 118
5.12 Évaluation des différents protocoles de combinaison d'embeddings de lemmes sur le Test avec le système CNN. 123
5.13 Évaluation des différents protocoles de combinaison d'embeddings de mots sur le Test avec le système CNN. 123
5.14 Statistiques par type de corpus, avec #voc : taille du vocabulaire, #occ : taille du corpus, et #1 : nombre des unités apparaissant une seule fois dans le corpus. Toutes les tailles sont reportées en *kilo (k)*. 126
5.15 Exactitude (A) du système CNN entrainé avec les embeddings polarisés, non polarisés and mixtes sur le corpus de test. 128
5.16 Statistiques sur les espaces d'embeddings de mots utilisés dans les versions statique (non mis à jour *màj*) et non statique du CNN (mis à jour *màj*), avec : # : le nombre de mots bougés dans l'espace, #+ : le nombre de mots polarisés positivement bougés, #− : le nombre de mots polarisés négativement bougés, et #± : le nombre de mots bougés qui sont positifs et négatifs à la fois 130
5.17 Statistiques sur les espaces d'embeddings de lemmes utilisés dans les versions statique (non mis à jour *màj*) et non statique (mis à jour *màj*) du CNN, avec : # : le nombre de lemmes bougés dans l'espace, #+ : le nombre de lemmes polarisés positivement bougés, #− : le nombre de lemmes polarisés négativement bougés, et # ± : le nombre de lemmes bougés qui sont positifs et négatifs à la fois 130

6.1 Récapitulation de l'impact des expériences phares réalisées. Les performances sont représentées par l'exactitude (A) 134

A.1 Performances du CNN sur le corpus de Dev avec longueur 300. 137
A.2 Évaluation des différents protocoles de combinaison d'embeddings de lemmes sur le Dev. 138
A.3 Évaluation des différents protocoles de combinaison d'embeddings de mots sur le Dev. 138

B.1 Exactitude des classifieurs RL et MLP sur *LABR_ar* dans le cadre d'une classification binaire 146
B.2 Exactitude des classifieurs RL et MLP sur *LABR_ar* dans le cadre d'une classification multi-classes 146
B.3 Comparaison des travaux effectués sur le corpus LABR dans le cadre classification multi-classes 147
B.4 Performances de la régression logistique sur les corpus *LABR_ar* et *LABR_en* . . . 147
B.5 Exemples en arabe dans corpus *LABR_ar* et leurs équivalents traduits en anglais dans *LABR_en* 148
B.6 Configurations et résultats des expériences réalisées pour élargir le corpus d'apprentissage 149

Liste des abréviations

AC	**A**rabe **C**lassique
AD	**A**rabe **D**ialectale
AO	**A**nalyse d'**O**pinions
AOA	**A**nalyse d'**O**pinions en **A**rabe
ASM	**A**rabe **S**tandard **M**oderne
CNN	**C**onvolutional **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**ime **M**emory
MLP	**M**ultilayer **P**erceptron
NB	**N**aif **B**ayes
RL	**R**égression **L**ogistique
RN	**R**éseau de **N**eurones
SVM	**S**upport **V**ector **M**achines
SP	**S**core de **P**olarité

Introduction

Avec la montée d'internet et la révolution des réseaux sociaux, un grand nombre d'individus peuvent exprimer leurs points de vue et leurs sentiments sur des entités, des produits, des personnes, *etc*. Cette croissance est accompagnée d'un énorme volume de données d'opinions disponibles sur le web. En effet, 2.5 milliards d'octets de données sont créés chaque jour. Ces dernières années, 90% des données dans le monde ont été générées [1].
Dans ce contexte, l'analyse automatique d'opinions connait un intérêt croissant de la part des entreprises et de la communauté scientifique [2], et la recherche en analyse d'opinions est un domaine en plein essor.

Le problème d'analyse d'opinions est complexe. Il regroupe plusieurs tâches telles que : la détection de la subjectivité, la détection de la polarité et son intensité, l'identification de l'entité sur laquelle porte l'opinion et ses différents aspects, détermination de la polarité par aspect, identification du détenteur d'opinion et de son profil, étude de l'évolution d'opinions sur une entité donnée en fonction du temps. En plus de la variété de tâches, s'ajoute la nature du support utilisé pour véhiculer les opinions. Nous distinguons 3 natures : énoncé textuel, oral et audiovisuel. Dans cette thèse, l'analyse d'opinions (AO) est réduite à la détection de la polarité d'un énoncé textuel donné.

Les travaux effectués dans le domaine de l'AO pour la détection de la polarité d'un énoncé textuel peuvent être classés selon trois approches. La première est symbolique, elle utilise des lexiques et des règles linguistiques. La deuxième consiste en une approche numérique qui s'appuie sur des méthodes d'apprentissage automatique et/ou profond. La troisième consiste en approche hybride qui est une combinaison des deux précédentes : elle utilise à la fois des lexiques et des algorithmes d'apprentissage automatique. Jusqu'à récemment, les machines à vecteurs de supports (SVM) et les classifieurs naïfs de Bayes (NB) représentaient les classifieurs les plus répandus dans ce domaine. Suivant la mouvance actuelle, la majorité des travaux récents fait recours à l'apprentissage profond et aux réseaux de neurones artificiels.
Dans cette thèse, nous nous focalisons sur la détection de polarité par des méthodes à base de réseaux de neurones pour la langue arabe.
Les avancées scientifiques récentes dans les techniques d'apprentissage profond ainsi que la croissance des puissances de calcul, ont mené à l'amélioration significative des performances dans différents domaines tels que la reconnaissance de la parole ou la traduction automatique. La recherche en analyse d'opinions a également tiré profit de l'apprentissage profond, et plusieurs travaux ont été réalisés avec ce type d'apprentissage mais peu concernant la langue arabe.

La langue arabe représente l'une des langues les plus répandues dans le monde. Trois catégories d'arabe peuvent être distinguées : l'arabe classique (AC) utilisé dans les textes religieux, l'arabe

1. https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1&sf100871281=1
2. https://trends.google.com/trends/explore?date=all&q=sentiment%20analysis

standard moderne (ASM) comme étant la langue officielle et, l'arabe dialectal (AD) utilisé par les loculeurs arabes dans leurs communications informelles de tous les jours. Les énoncés textuels sont principalament écrits en ASM et AD. Nous nous limitons notre étude à ces deux catégories d'arabe.
Un mot en arabe est défini, au sens graphique, comme étant une séquence de caractères délimitée par deux séparateurs (blanc ou autre marqueur de séparation, tel que la ponctuation). Or, la langue arabe est caractérisée par son agglutination et sa richesse morphologique. Ces caractéristiques apparaissent dans la composition des mots et conduisent à un vocabulaire éparse. Une phrase en arabe peut être composée d'un seul mot, ce qui reflète la complexité des mots arabes.

Pour le traitement automatique du langage naturel écrit, la majorité des réseaux neuronaux prennent comme entrée des représentations vectorielles continues (*embeddings*) de mots. L'espace de projection est un espace continu supposé préserver les similarités sémantiques et syntaxiques des mots. Les embeddings de mots se sont révélés être un atout fondamendal pour plusieurs tâches de traitement du langage naturel, y compris l'analyse d'opinions. La complexité de la langue arabe peut impacter la qualité de l'espace d'embeddings.
Actuellement, les embeddings pré-entrainés existants représentent un mot arabe sans considération des caractéristiques d'agglutination et de la richesse morphologique de l'arabe. Nous proposons dans cette thèse de construire des embeddings spécifiques à l'arabe et de les évaluer pour la tâche d'analyse d'opinions.

Ce document est organisé en deux grandes parties. Nous présentons dans un premier temps l'état de l'art de l'analyse d'opinions, les caractéristiques de la langue arabe et une étude de l'existant en analyse d'opinions en arabe (AOA). Nous développons ensuite le travail réalisé durant cette thèse, à savoir l'élaboration de systèmes d'AOA et l'étude de différents embeddings de mots existants, la proposition des embeddings spécifiques à l'arabe et l'amélioration des performances, et enfin l'évaluation des embeddings spécifiques, de façon intrinsèque et extrinsèque, dans le cadre de l'analyse d'opinions.

Nous présentons de façon générale le domaine d'analyse d'opinions dans le premier chapitre : définition, origine, domaines d'application ainsi qu'une synthèse de la littérature sur le domaine de l'AO.

Dans le deuxième chapitre, nous présentons la langue arabe et nous mettons l'accent sur certaines de ses spécificités qui sont en liaison directe avec la tâche d'analyse d'opinions.

Nous présentons, dans le chapitre 3, les réseaux de neurones les plus utilisés ainsi que les techniques de construction de représentations vectorielles continues. Nous présentons également l'état de l'art des travaux réalisés en AOA utilisant des réseaux de neurones artificiels.

Mes contributions s'étalent sur les chapitres 4 et 5. Nous proposons, dans le chapitre 4, un protocole expérimental pour l'analyse d'opinions en arabe. Nous menons également une étude qualitative sur les embeddings de mots pré-entrainés existants.

En arabe, les embeddings existants représentent les mots. Or, un mot arabe est complexe. Pour cette raison, nous proposons, dans le chapitre 5, des embeddings spécifiques à l'arabe prenant en compte l'agglutination et la richesse morphologique de cette langue. Nous proposons également,

suite aux résultats de classification, d'étudier et d'évaluer différentes approches pour combiner les entrées (embeddings spécifiques) et les sorties des systèmes afin d'améliorer les résultats. Nous finissons par une analyse qualitative basée sur des méthodes intrinsèque et extrinsèque. Pour l'évaluation intrinsèque des embeddings, nous introduisons la notion de la stabilité de polarités (*sentiment stability* SS) dans les espaces d'embeddings. En ce qui concerne l'évaluation extrinsèque, nous évaluons les performances du réseau convolutif CNN, dans sa version statique et non statique, avec les embeddings spécifiques à l'arabe. La version non statique consiste à mettre à jour les embeddings lors de l'apprentissage du CNN à la tâche de la détection de la polarité. Elle permet d'obtenir des embeddings spécifiques à la tâche. Par opposition à la version non statique du CNN, la version statique ne met pas à jour les embeddings.

Finalement, une conclusion des points clés de la thèse est présentée dans le dernier chapitre, ainsi que quelques perspectives pour de futurs travaux de recherche.

Chapitre 1

Analyse d'opinions

Sommaire

1.1 Notion d'analyse d'opinions 6
1.1.1 Étymologie 6
1.1.2 Modélisation formelle d'opinions 6
1.1.3 Types d'opinions 8
1.1.4 Niveaux d'analyse d'opinions 9
1.2 Approches d'analyse d'opinions 10
1.2.1 Approche symbolique 11
1.2.2 Approche numérique 14
1.2.3 Approche hybride 17
1.3 Challenges de l'analyse d'opinions en arabe 18
1.3.1 Manque de ressources 18
1.3.2 Opinion spam 18
1.3.3 Ironie et sarcasme 18
1.4 Domaines d'application de l'analyse d'opinions 19
1.5 Conclusion 20

Le domaine d'analyse d'opinions a connu récemment un intérêt grandissant. La croissance rapide de ce domaine de recherche coïncide avec la prolifération du web 2.0 et surtout des médias sociaux (des forums, des blogs, des réseaux sociaux, *etc*).

Ce chapitre présente, d'une façon générale, le domaine de recherche en analyse d'opinions. Nous introduisons, dans un premier temps, la notion d'analyse d'opinions. Nous en présentons l'origine et l'étymologie et définissons formellement cette problèmatique. Nous présentons également les différents types et niveaux d'analyse. Nous élaborons, dans un deuxième temps, une synthèse de la littérature sur le domaine de l'analyse d'opinions. Nous finissons par présenter les différents domaines d'applications de l'analyse d'opinions.

1.1 Notion d'analyse d'opinions

L'analyse d'opinions, dans le domaine informatique, s'intéresse au traitement automatique des opinions, des sentiments, et de la subjectivité exprimés ou véhiculés dans des énoncés textuels et audiovisuels. Les opinions concernent des entités qui peuvent être des produits, des services, des thématiques, des personnes publiques, des organisations, *etc*. Les énoncés textuels peuvent être présentés selon différents formats/types : article dans un journal, commentaire/critique dans un site web, post/commentaire dans des réseaux sociaux (facebook, twitter, *etc.*). Les énoncés oraux, présentés dans des documents audiovisuels, sont aussi présentés sous différents formats : journaux télévisés, émissions radio, vidéos youtube, *etc*. Cette thèse s'intéresse aux énoncés textuels de type commentaire/critique sur des sites web.

Dans cette section, nous présentons le domaine d'analyse d'opinions : étymologie, modélisation formelle, types et niveaux d'analyse d'opinions.

1.1.1 Étymologie

Dans la littérature, les deux termes *analyse d'opinions* et *analyse de sentiments* désignent le même domaine d'étude et sont utilisées de façon interchangeable. Le domaine d'analyse d'opinions se trouve sous différentes expressions : analyse de sentiments ou d'opinions en français, et *sentiment analysis* ou *opinion mining* en anglais. Dans cette thèse, nous utilisons le terme *analyse d'opinions* (AO) pour faire référence à cette thématique de recherche.

D'un point de vue historique en analyse d'opinions, le terme *sentiment* est apparu avec [Das & Chen, 2001]. Et le terme *opinion* est apparu avec [Tong, 2001]. Le terme *sentiment* est défini comme *"la perception du corps réel modifié par l'émotion"* par William James [James, 1884] et comme *"la composante de l'émotion qui implique les fonctions cognitives de l'organisme, la manière d'apprécier"* sur Wikipédia. Alors que le terme *opinion* est défini comme *"un jugement que l'on porte sur un individu, un être vivant, un phénomène, un fait, un objet ou une chose"* sur Wikipédia.

1.1.2 Modélisation formelle d'opinions

Le problème de l'analyse d'opinions est complexe. Plusieurs tâches peuvent être répertoriées et varient selon le domaine et le cadre applicatif. Nous distinguons principalement les tâches suivantes :

- Détection de la subjectivité : distinction entre faits et opinions
- Détection de la polarité : détermination de la polarité/axiologie de l'opinion (positive, négative ou éventuellement neutre)
- Détection de l'intensité de la polarité : L'intensité montre le degré de positivité ou de négativité, et varie de faible à forte.
- Identification de l'entité : détermination de l'entité ou l'objet sur lequel porte l'opinion
- Identification de la polarité par aspect : détermination des aspects décrivant une entité donnée et de la polarité associée à chaque aspect

— Identification du détenteur de l'opinion : détermination de la personne qui a exprimé cette opinion

Pour mieux comprendre chacune de ces tâches, nous considérons les exemples (Commentaire A), (Commentaire B) et (Commentaire C) suivants issus du site web amazon :

Commentaire A :
Samsung dévoile sa nouvelle gamme de téléphones portables S10.

Commentaire B :
Le S9 est vraiment bluffant : écran superbe, appareil photo magnifique, téléphone fluide... Bref très bon achat.

Commentaire C :
J'ai acheté un S9 les six derniers mois. J'ai trop aimé. Par contre, ma femme n'est pas satisfaite de la qualité de son appareil photo.

La détection de la subjectivité consiste à distinguer les faits des opinions, ce qui revient à distinguer les énoncés objectifs des énoncés subjectifs. Le commentaire A est un énoncé objectif (fait). Par contre, les commentaires B et C sont des énoncés subjectifs (opinions).
La détection de polarité représente le cœur de l'analyse d'opinions. Elle consiste à étudier l'opinion afin de déterminer sa polarité : positive ou négative (ou éventuellement neutre). Par exemple, la polarité du commentaire B est positive.
La tâche d'extraction d'entité et/ou aspects consiste à déterminer l'entité et éventuellement ses aspects sur lesquels porte l'opinion. Pour le commentaire B, l'entité est le téléphone S9 et les aspects sont écran et appareil photo. La tâche d'analyse d'opinions par aspect consiste à déterminer la polarité pour chaque aspect de l'entité : la polarité positive pour l'aspect *écran* portée par le mot *superbe* et la polarité positive pour l'aspect *appareil photo* portée par le mot *magnifique*.
En plus de l'extraction des entités/aspects et la détection de polarité, il est parfois intéressant de préciser le détenteur de l'opinion. Le commentaire C contient les opinions de deux personnes : l'auteur du commentaire et sa femme.

Toutes ces tâches permettent de mieux comprendre et cerner le problème d'analyse d'opinions. Abstraction de toute tâche, [Hu & Liu, 2004a] propose une structure générale qui permet d'unifier plusieurs axes de recherche et concevoir des solutions robustes. En effet, toute personne exprime son opinion sur une entité ou un objet quelconque : elle donne son point de vue global sur l'entité et peut éventuellement détailler ou expliquer son opinion globale en donnant de points de vue partiels et spécifiques à certaines caractéristiques de l'objet. L'opinion est donc portée sur une entité bien précise : c'est le concept central de l'opinion. Cette entité peut être décrite par un nombre arbitraire

d'aspects : une polarité par aspect. Ainsi, l'opinion (O) est définie par le quintuple 1.1 :

$$O = (e,\ a,\ p,\ d,\ t) \tag{1.1}$$

avec :

- e : l'entité de l'opinion
- a : un aspect particulier de l'entité
- p : la polarité relative à l'aspect a
- d : le détenteur de l'opinion
- t : le temps auquel l'opinion est valide.

Dans le cas où le détenteur d donne un point de vue général sur l'entité e sans détailler ses différents aspects, l'aspect spécial GENERAL est utilisé pour dénoter l'élément a dans la formule 1.1. Ce qui signifie que les deux éléments e et a représentent ensemble l'entité de l'opinion. Lorsque l'aspect a est bien défini et différent de GENERAL, l'analyse d'opinions est connue par *analyse d'opinions basée sur les aspects* [Hu & Liu, 2004a] : c'est une analyse plus fine des caractéristiques de l'entité. Cet élément a semble nécessaire principalement pour les applications d'AO dédiées aux produits et services (restauration, hôtelerie, *etc*).
L'information sur le détenteur de l'opinion d et le temps de publication t ne sont pas utiles pour certaines applications. L'extraction du détenteur d'opinion est hors de propos quand on s'intéresse à l'opinion collective sur une entité. Le temps d'opinion peut être important dans le cas d'une étude temporelle de l'évolution de l'opinion, ce qui peut être intéressant dans les domaines de la politique et du marketing. Par conséquent, c'est la nature de l'application et son domaine qui définissent la liste des éléments de l'opinion O à considérer pour répondre à ses fins.

La définition 1.1 définit l'opinion de façon générale, mais dans la pratique, de nombreuses applications se concentrent sur la polarité p sans se soucier des différentes caractéristiques ou composantes de l'entité. L'objectif de cette thèse consiste à déterminer la polarité globale d'un énoncé textuel sans détailler les polarités par aspects.

1.1.3 Types d'opinions

L'opinion peut être classée en différents types selon les traits d'explicité et de régularité. En effet, l'explicité distingue l'opinion explicite et implicite. Alors que la régularité permet de distinguer l'opinion régulière de celle comparative.

Selon le trait d'explicité, l'opinion est exprimée directement sur l'entité ou un aspect quelconque de l'entité dans l'énoncé textuel. Par exemple, le commentaire B est direct puisqu'il décrit de façon explicite l'entité S9 et lui associe la polarité positive. En opposition à l'opinion explicite, l'opinion implicite est exprimée de façon indirecte. C'est le cas du commentaire D : l'internaute était déçu de la qualité du service de livraison, il exprime sa déception de façon implicite via le retard de réception de sa commande.

Commentaire D :
J'ai reçu ma commande Amazon avec 15 jours de retard.

Commentaire E :
Le S9 est meilleur qu'un iphone. Il contient plus d'options...

Selon le trait de régularité, une opinion peut être évaluée de deux façons différentes : régulière et comparative. L'opinion régulière est un point de vue simple et direct. Elle porte sur une seule entité sans mentionner d'autres entités similaires. Les commentaires B et C sont des opinions régulières. Alors que l'opinion comparative contient une comparaison entre plusieurs entités similaires. Par exemple, le commentaire E exprime une comparaison entre les deux entités S9 et iphone. Les opinions comparatives sont plus difficiles à analyser. La difficulté consiste à déterminer, parmi les entités comparées, les entités préférées par les détenteurs d'opinions. Par exemple, le S9 représente l'entité préférée du détenteur du commentaire E.

Les opinions implicites et comparatives sont plus compliquées que celles explicites et régulières. Elles nécessitent plus d'effort d'analyse.

1.1.4 Niveaux d'analyse d'opinions

Nous nous intéressons, dans cette thèse, à l'analyse d'opinions dans les énoncés textuels. La longueur d'un énoncé varie selon la source et le détenteur. D'une part, chaque source exige un ensemble de règlements entre autres la longueur du commentaire. Par exemple, les tweets ne doivent pas dépasser 140 caractères, alors que la longueur des commentaires peut être grande dans d'autres sites web, comme par exemple facebook. D'autre part, certaines personnes (détenteurs) ont tendance à être concis et bref dans leurs commentaires, et d'autres expliquent et détaillent leurs opinions. Les règlements de la source et la personnalité du détenteur interviennent ainsi dans l'énoncé textuel et déterminent sa longueur.

L'opinion peut donc s'analyser selon différents niveaux de granularité. Le premier niveau est global et présent à l'échelle du document, le deuxième est local et porté par la phrase, le troisième niveau est plus restreint et véhiculé dans un segment ou groupe de mots, et le dernier niveau est porté par un seul mot. Les quatre niveaux d'analyse ainsi distingués sont détaillés dans la suite.

1.1.4.1 Niveau document

La tâche d'analyse d'opinions au niveau document consiste à associer au document en entier une polarité : positive ou négative (et parfois neutre) [Pang *et al.*, 2002; Turney, 2002]. Ce niveau d'analyse suppose que l'opinion exprimée dans le document est liée à une seule entité (un commentaire sur film par exemple). Ceci signifie implicitement que cette analyse ne s'applique pas aux documents qui évaluent ou comparent plusieurs entités en même temps. Cela implique aussi que si différentes polarités sont exprimées, il faudra choisir la plus pertinente.

1.1.4.2 Niveau phrase

Ce niveau d'analyse est étroitement lié à la classification de subjectivité. Ce qui signifie que l'analyse d'opinions distingue les phrases objectives des phrases subjectives, c'est-à-dire différencier entre les faits et les opinions. Une phrase objective exprime des informations factuelles alors qu'une phrase subjective exprime des points de vue et des opinions subjectives. Les phrases subjectives peuvent ensuite être classées en polarité positive ou négative [Liu, 2012].

1.1.4.3 Niveau aspect

Les deux analyses au niveau document et phrase ne précisent pas exactement ce que les personnes aiment ou détestent. L'analyse au niveau aspect est plus fine. Cette granularité s'intéresse à un aspect donné et détermine la polarité relative à cet aspect. Il s'agit donc d'attribuer une polarité (positive, negative ou neutre) à chaque aspect évoqué dans un document d'opinion [Pontiki *et al.*, 2014]. Ceci nécessite, dans un premier temps, une extraction d'aspects et une identification de polarité pour chaque aspect dans un deuxième temps. Par exemple, les aspects présents dans le commentaire B sont écran et appareil photo et l'analyse de ce commentaire permet d'identifier la polarité pour chaque aspect : (écran, positive) et (appareil photo, positive).

1.1.4.4 Niveau mot

Le niveau mot représente le niveau le plus bas. L'analyse d'opinions au niveau mot consiste à déterminer la polarité d'un mot. Cette tâche permet notamment la construction de lexique de mots polarisés. Ce dernier est composé d'un ensemble de couples (ω, *pol*), où *pol* est un score (une valeur réelle) reflétant la polarité du mot ω.
Les mots polarisés sont principalement des adjectifs et des adverbes, mais ils peuvent aussi être des verbes et des noms. Ces mots codent des états émotionnels désirable ou indésirable : les mots qui codent un état désirable (par exemple : beau, magnifique, heureux) ont une polarité ou orientation positive, tandis que ceux qui codent un état indésirable (par exemple : mauvais, terrible, décevant) ont une orientation négative.

Les lexiques polarisés sont généralement utilisés pour l'analyse d'opinion de niveau plus haut (document, phrase ou aspect). Nous détaillons, dans la section 2.3.1 du chapitre 2, les méthodes de construction de lexiques dans le cadre spécifique de la langue arabe.

1.2 Approches d'analyse d'opinions

La recherche en analyse d'opinions est très active en TALN depuis la fin des années 90. [Kaur & Gupta, 2013; Hussein, 2018; Verma & Thakur, 2018] présentent un survol de l'état de l'art de ce domaine de recherche.
Afin de réaliser ces recherches, plusieurs types de ressources existent : des corpus annotés [Al-Ayyoub

et al., 2019], des lexiques de mots polarisés [Tong, 2001; Turney, 2002; Turney & Littman, 2003; Benamara *et al.*, 2007; Taboada *et al.*, 2011], des ontologies [Esuli & Sebastiani, 2006], *etc*.

Plusieurs travaux ont été menés afin de résoudre le problème d'analyse d'opinions avec différentes méthodes (linguistique et/ou numérique). Ces travaux peuvent être donc classés selon trois approches. La première est symbolique, elle utilise des lexiques et des règles linguistiques. La deuxième consiste en une approche numérique qui s'appuie sur des méthodes d'apprentissage automatique. Pour finir, il existe une approche hybride qui est une combinaison des deux précédentes : elle utilise à la fois des lexiques et des algorithmes d'apprentissage automatique. La figure 1.1 présente un panorama de ces approches. Nous présentons, dans la suite, les trois approches.

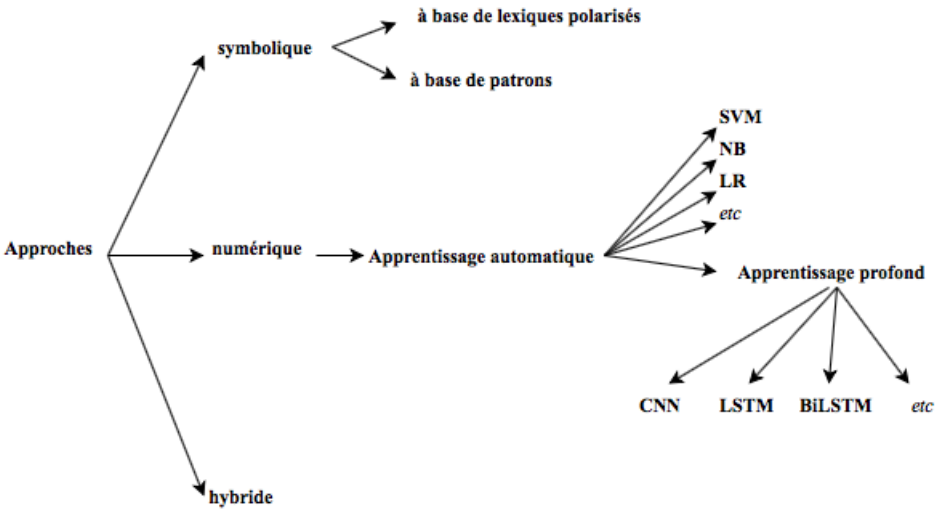


FIGURE 1.1 – Les approches en analyse d'opinions.

1.2.1 Approche symbolique

Pour déterminer la polarité d'un énoncé textuel, l'approche symbolique en analyse d'opinions se base sur des règles linguistiques et des lexiques polarisés. Elle repose simplement sur l'idée que la polarité d'un texte est obtenue en fonction de l'orientation sémantique (dite aussi polarité) des mots qui composent ce texte. L'élément fondamental dans cette approche est le lexique polarisé. Ce dernier est indispensable pour toute méthode symbolique (dite aussi linguistique) d'analyse d'opinions.
Un lexique polarisé comporte un ensemble de mots ayant des traits spécifiques marquant la polarité (positive ou négative, ou parfois neutre). Il s'agit pour la plupart de verbes et d'adjectifs, mais aussi de noms communs et d'adverbes. Ces catégories syntaxiques portent les mots polarisés quelle que soit la langue. Le tableau 1.1 reporte un ensemble de mots polarisés classés par catégorie syntaxique dans les langues anglais, français et arabe. Ce classement se base sur le sens propre de mots. Néanmoins, plusieurs mots peuvent admettre des polarités différentes dans des contextes différents [Xia *et al.*, 2015]. Par exemple, un mot polarisé positivement selon son sens propre peut devenir polarisé négativement s'il est utilisé dans le cadre d'une figure de style de type opposition (e.g. antiphrase,

antithèse, *etc*).

Catégorie syntaxique	**Polarité**	**Langue**	**Exemples**
Adjectif	positif	En	beautiful, happy, great, *etc*
		Fr	beau, heureux, magnifique, *etc*
		Ar	جميل, سعيد, *etc*
	négatif	En	sad, bad, ugly, *etc*
		Fr	triste, mauvais, ennuyeux, *etc*
		Ar	حزين, تعيس, *etc*
Verbe	positif	En	love, like, prefer, *etc*
		Fr	aimer, adorer, apprécier, *etc*
		Ar	يحب, يعجب, يحبذ, *etc*
	négatif	En	dislike, hate, decline, *etc*
		Fr	détester, déprécier, insupporter, *etc*
		Ar	يكره, يحزن, يشتم, *etc*
Nom	positif	En	love, like, *etc*
		Fr	amour, plaisir, bonheur, *etc*
		Ar	حبّ, فرح, متعة, *etc*
	négatif	En	damage, dirty, fail, *etc*
		Fr	tristesse, inquiétude, chagrin, *etc*
		Ar	كره, جهل, علقم, *etc*

TABLE 1.1 – Exemples de mots polarisés classés selon leurs catégories syntaxiques dans les langues anglais (En), français (Fr) et arabe (Ar).

Plusieurs travaux ont été menés en faisant appel à cette approche. La majorité des travaux en approche symbolique commencent par une construction de lexiques polarisés comme ressource lexicale et finissent par une évaluation en polarité d'un texte donné. Nous présentons, dans la suite, ces deux phases, à savoir a) la construction de lexiques polarisés et b) l'évaluation en polarité.

a) Construction de lexiques polarisés :

Les lexiques polarisés représentent une ressource lexicale importante pour les méthodes symboliques. Lorsqu'elle n'est pas disponible, une construction de cette ressource devient nécessaire. Un lexique polarisé peut être construit de trois façons différentes : manuelle, semi-automatique ou automatique. Ces dernières seront présentées dans le chapitre 2.

Les premiers travaux en construction de lexiques polarisés considèrent les adjectifs comme indicateurs primordiaux dans la polarité du texte [Hu & Liu, 2004a; Taboada *et al.*, 2011]. La robustesse du lexique dépend principalement de son domaine d'application. En effet, certains mots changent de polarité d'un domaine à un autre. Il n'est donc pas évident de construire un lexique unique pour les différents domaines. Néanmoins, il existent des mots universels dont la polarité ne change pas quelque soit le domaine d'application [Pupier, 1998]. À ce titre, le tableau 1.2 cite quelques mots universels dont la polarité ne change pas en fonction du domaine.

Catégorie syntaxique	Polarité	Mot universel
Adjectif	positive	beau, magnifique, fabuleux, *etc*
	négative	mauvais, néfaste, ennuyeux, *etc*
Nom	positive	amour, joie, bonheur, *etc*
	négative	haine, tristesse, mélancolie, *etc*
Verbe	positive	aimer, adorer, apprécier, conseiller, *etc*
	négative	détester, déconseiller, déprécier, *etc*
Adverbe	positive	heureusement, appréciablement, tendrement, *etc*
	négative	malheureusement, amèrement, mélancoliquement, *etc*

TABLE 1.2 – Exemples de mots universels polarisés classés selon leurs catégories syntaxiques.

En plus de ces mots à polarité *a priori*, il existe des mots dont la polarité peut changer selon le contexte [Riloff & Wiebe, 2003; Xia *et al.*, 2015]. Il s'agit principalement de mots polysémiques ou bien des homonymes ayant des polarités différentes. La polarité d'un mot non-polysémique peut également changer à l'intérieur d'un même domaine, selon l'entité (ou éventuellement l'aspect) qu'il décrit. Par exemple, pour un ordinateur portable, une batterie "*grande*" est un point négatif, mais un écran "*grand*" est un point positif. La polarité des mots peut aussi dépendre de l'idéologie de l'auteur (le détenteur de l'opinion) et de ses préférences. Les textes politiques sont justement sensibles à ce fait. Par exemple, le mot "*bourgeois*" est fondé sur une sémantique neutre mais quand il s'agit de préjugé ou d'opinion, ce qui est "*bourgeois*" est souvent mal vu. Pour un énoncé textuel donné, le contexte de ses mots et l'idéologie de son auteur (détenteur) sont des facteurs à prendre en considération dans l'évaluation en polarité de l'énoncé textuel.

b) Évaluation en polarité :

L'évaluation en polarité représente le cœur de la méthode symbolique. Elle consiste à associer une polarité à un énoncé textuel donné en se basant sur les polarités des phrases qui composent l'énoncé. Elle suit, dans l'ordre, les trois étapes suivantes :

1. Diviser l'énoncé en phrases.
2. Déterminer la polarité de chaque phrase par identification et agrégation des mots polarisés, ou bien par patrons d'extraction de polarité :
   - **Identification et agrégation** :
     - **Identification** : elle consiste à identifier les mots polarisés qui existent dans la phrase en se basant sur un lexique polarisé.
     - **Agrégation** : elle consiste à agréger les mots polarisés de la phrase. La solution la plus simple consiste à compter le nombre de mots positifs et le nombre de mots négatifs. S'il y a une majorité de termes positifs, la phrase est déclarée positive. A l'inverse, si les mots négatifs sont plus nombreux, la phrase est déclarée négative. Les phrases possédant autant de mots négatifs que de mots positifs peuvent être considérées neutres. Si les entrées du lexique polarisés sont représentées par des scores, alors l'agrégation

consiste à calculer la somme des scores et conclure selon le signe de la somme. Si le signe de la somme est positif, alors la polarité associée est positive, et sinon la polarité négative est associée à la phrase.

Ce procédé est valide lorsque les phrases sont affirmatives et ne contiennent pas des termes de négation. Cependant dans les phrases négatives, il est important de considérer les termes de négation qui interviennent dans la détermination de la polarité de la phrase. En effet, l'ajout d'un terme de négation à une phrase affirmative inverse sa polarité. Certains travaux utilisent des règles linguistiques de type arbre syntaxique [Wilson *et al.*, 2005] ou arbre de dépendance [Nakagawa *et al.*, 2010]. En partant des feuilles, un calcul de polarité s'effectue en parcourant l'arbre de bas (les feuilles) en haut (la racine). Un calcul de polarité se fait à chaque niveau de l'arbre pour obtenir des polarités intermédiaires relatives aux segments constituant la phrase.

— **Patrons d'extraction de polarité** :
ils peuvent être utilisés pour déterminer la polarité d'une phrase. Ils sont principalement utilisés lorsque le domaine d'étude est caractérisé par un style d'écriture et un champs lexical bien définis (par exemple, le domaine juridique, médical, *etc.*). La méthode basée sur des patrons d'extraction est longue et coûteuse. Ces patrons nécessitent souvent un nettoyage manuel par des experts avant d'être réellement exploitables.

3. Associer une polarité à l'énoncé en se basant sur les polarités des phrases qui le composent.

Les travaux selon l'approche symbolique nécessitent un lexique polarisé. La construction de ce dernier est couteuse : elle nécessite du temps et des experts humains maitrisant la langue. Un lexique polarisé est caractérisé par sa couverture est sa qualité. Selon l'aspect couverture, le lexique doit contenir tous les mots polarisés possibles. Et selon l'aspect qualité, les mots du lexique doivent avoir les polarités (ou intensités de polarité) exactes. Cependant, chaque langue possède ses propres caractéristiques et les mots changent de polarité selon le contexte et le domaine utilisés. Ceci signifie qu'un lexique générique n'est pas fiable pour tous les domaines d'application d'analyse d'opinions. La construction de lexique par domaine semble être une solution, c'est-à-dire construire un lexique pour le domaine d'hôtellerie, un autre pour la restauration, *etc*.

1.2.2 Approche numérique

Dans le cas où l'analyse d'opinions consiste à déterminer la polarité d'un texte, le problème peut être considéré comme un problème de classification selon la polarité. L'approche numérique repose sur des techniques d'apprentissage automatique et/ou profond. Ces méthodes se composent de deux étapes : apprentissage et prédiction. L'étape d'apprentissage consiste à construire un modèle à partir d'un corpus d'apprentissage (un ensemble d'exemples annotés en polarité). Alors que l'étape de prédiction utilise le modèle d'apprentissage entrainé pour attribuer une classe (une polarité) à un nouveau commentaire non étiqueté. Ainsi l'approche numérique traite les textes en entier et attribue une polarité et attribue une polarité globale à la fin de l'apprentissage du classifieur, tandis que l'approche

symbolique fait l'analyse du texte par mot ou groupe de mots véhiculant des porarités et agrège à la fin la polarité globale du texte.
Les méthodes numériques nécessitent un corpus annoté relatif à la tâche en question. Pour l'analyse d'opinions, le corpus d'apprentissage représente un ensemble d'exemples annotés en polarité.
Les méthodes classiques d'apprentissage automatique utilisent généralement la représentation vectorielle de type *sac de mots*, dit aussi *Bag of words* (BOW) pour représenter les documents. La représentation *sac de mots* tient compte uniquement de la fréquence d'apparition des mots sans prendre en considération leurs ordres dans les phrases.
Les méthodes d'apprentissage profond utilisent des vecteurs denses de type *embedding* pour représenter les mots et les documents. L'embedding encode chaque terme d'une phrase en un vecteur numérique tout en respectant une notion de proximité sémantique au sens où deux termes similaires seront encodés par des vecteurs proches.
Nous présentons succinctement les techniques d'apprentissage automatique et profond dans la suite.

1.2.2.1 Apprentissage automatique

L'apprentissage automatique consiste à entrainer un classifieur en se basant sur des descripteurs, dits aussi traits ou fonctions caractéristiques (*features* en englais), spécifiques à la tâche d'analyse d'opinions. Ces descripteurs permettent d'inférer pour un nouvel énoncé sa polarité [Al-Ayyoub *et al.*, 2019]. Les bonnes performances des classifieurs sont conditionnées d'une part par la quantité de données d'apprentissage et d'autre part par la qualité des descripteurs. En effet, la taille du corpus d'apprentissage doit être suffisante pour l'entraînement du classifieur, et les descripteurs doivent être spécifiques à la tâche.

Plusieurs travaux en analyse d'opinions sont basés sur des techniques d'apprentissage automatique. Les classifieurs utilisés dans ce domaine sont principalement : l'algorithme naïf de Bayes (NB), les machines à vecteurs de support (SVM), les k plus proches voisins (KNN), la régression logistique (LR) et l'entropie maximale (ME). Quelque soit sa nature, l'algorithme d'apprentissage automatique (classifieur) nécessite que le texte soit représenté par un ensemble de descripteurs. Le texte à classer est alors représenté un vecteur à valeurs discrètes. Chaque élément du vecteur représente la valeur d'un descripteur. Les descripteurs les plus utilisés dans la littérature peuvent être classés en trois grandes catégories : surfacique, syntaxique, et lexicale. Le tableau 1.3 résume les différents descripteurs.

Le choix des descripteurs est très important dans la qualité du modèle d'apprentissage. Généralement, la détermination des attributs pertinents se fait par un algorithme de sélection d'attributs. La performance du classifieur varie d'un jeu d'attributs à un autre. Ces attributs demandent une bonne conception et une véritable réflexion pour définir ou deviner les attributs adéquats à la tâche de classification. Le SVM s'est montré le modèle à base de caractéristiques prédéfinies le plus performant pour la tâche d'analyse d'opinions [Bologna & Hayashi, 2018; Saxena *et al.*, 2019].
Nous détaillons, dans la suite, le principe des classifieurs les plus utilisés.

Catégorie	Descripteurs d'apprentissage
Surfacique	Signes de ponctuation : la presence ou l'absence de différents signes de ponctuation et principalement le point d'exclamation ou le point d'interrogation, le nombre de sequences continues de points d'exclamation, de points d'interrogation et les deux
	Émoticônes : la présence ou l'absence d'une émoticône typographique
Lexicale	le nombre de mots positifs, négatifs, la polarité du premier et du dernier mot
	le nombre d'occurence de mots (sac de mots), de n-grammes de caractères
	la présence ou l'absence de termes de négation
Syntaxique	le nombre de quelques étiquettes morpho-syntaxiques de mots (par exemple, les adjectifs et les adverbes)
	la présence de patrons syntaxiques discriminants

TABLE 1.3 – Exemples de descripteurs utilisés dans les techniques d'apprentissage automatique.

- Machine à vecteurs de support :

Les machines à vecteurs de supports , dit aussi *Support Vector Machine* en anglais (SVM), représentent une méthode de classification binaire introduite par [Vapnik, 1982]. Cette méthode est linéaire, elle suppose l'existence d'un hyperplan. Le but du SVM est de trouver un hyperplan qui va séparer deux nuages de points représentant les deux classes positive ou négative en maximisant la marge (la distance du point le plus proche a l'hyperplan) entre ces deux classes.

- Naive Bayes :

Naive Bayses (NB) est une méthode probabiliste basée sur le théorème de Bayes avec une forte indépendance des hypothèses (d'où le terme *naïve*) [Rish *et al.*, 2001]. Cette méthode linéaire requiert relativement peu de données d'entraînement pour estimer les paramètres nécessaires à la classification, à savoir moyennes et variances des différentes variables.

- Arbre de décision :

L'arbre de décision est une méthode de classification binaire qui consiste à partitionner les données en sous-ensembles les plus purs possible [Stone, 1984]. L'arbre modélise une hiérarchie de tests sur les valeurs d'un ensemble de descripteurs. Il décide une classe (par exemple positive, négative) en fonction des valeurs de descripteurs.

- Régression logistique

La régression logistique (RL) représente une méthode de classification binaire [Desjardins, 2005]. Elle permet d'étudier le lien entre la variable expliquée qualitative (les classes) et les variables explicatives quantitatives (les descripteurs). La régression logistique multinomiale est une extension de la RL binaire et s'applique lorsque le nombre de classes est supérieur à 2.

1.2.2.2 Apprentissage profond

L'apprentissage profond consiste à utiliser des réseaux de neurones artificiels à plusieurs couches pour entraîner un modèle de classification. Un réseau de neurones est en effet capable de déterminer

de lui-même les attributs pertinents à la tâche. Il parvient à trouver l'information utile et les attributs discriminants pour la classification. Pour cela, il doit disposer d'un très grand nombre d'exemples d'apprentissage hétérogènes et représentatifs pour une bonne classification.

Plusieurs architectures neuronales ont été utilisées pour la tâche d'analyse d'opinions. Parmi les plus répandues, on cite à titre d'exemple : les réseaux de neurones convolutifs (CNN), les réseaux récurrents de type LSTM ou BiLSTM [Kim, 2014b; Croce *et al.*, 2016; Tai *et al.*, 2015; Dai & Le, 2015; Zhou *et al.*, 2016].

L'espace des embeddings est un espace continu supposé préserver les similarités sémantiques et syntaxiques des mots. Les embeddings de mots se sont révélés être un atout fondamendal pour plusieurs tâches de traitement du langage naturel, y compris l'analyse d'opinions.
Word2vec [Mikolov *et al.*, 2013b] et Glove [Pennington *et al.*, 2014] sont parmi les algorithmes de construction d'embeddings les plus répandus. Plus récemment, les embeddings contextuels *Elmo* [Peters *et al.*, 2018] et BERT [Grave *et al.*, 2018] sont apparus pour gérer à la fois les contextes linguistiques et la syntaxe/sémantique des mots.
En plus des embeddings standards, [Tang *et al.*, 2014] ont étendu l'algorithme de [Collobert *et al.*, 2011a] pour l'apprentissage des embeddings de mots et ont développé trois réseaux de neurones pour apprendre les embeddings spécifiques au sentiment [Tang *et al.*, 2016].

Nous présentons, dans le chapitre 3, les réseaux de neurones les plus utilisées en analyse d'opinions, les modèles de construction d'embeddings, ainsi que les travaux réalisés avec ces réseaux pour l'analyse d'opinions en arabe.

1.2.3 Approche hybride

L'approche hybride est une combinaison des deux approches symbolique (section 1.2.1) et numérique (section 1.2.2) [Prabowo & Thelwall, 2009]. Elle utilise à la fois des méthodes d'apprentissage automatique ou profond et des lexiques polarisés ou patrons linguistiques. L'approche hybride consiste à combiner les méthodes utilisées dans les approches symbolique et numérique. Elle consiste donc à utiliser des lexiques polarisés et des techniques d'apprentissage automatique. L'hybridation la plus utilisée consiste à considérer, pour la construction de modèle numérique, des descripteurs liés aux mots polarisés : la présence ou le nombre d'occurence ou score de mots positifs et négatifs (voir les descripteurs lexicaux mentionnés dans le tableau 1.3). Une autre façon hybridation consiste à utiliser en cascade un classifieur numérique et un autre à base de lexique.
Différentes sortes de combinaison sont possibles : séquentielle, parallèle ou pondérée. La combinaison séquentielle consiste à appliquer par exemple un classifieur à base de lexique dans un premier temps et un classifieur statistique dans un deuxième temps (ou inversement). La difficulté de cette combinaison réside dans l'ordre de classifieurs : quel classifieur choisir pour l'appliquer en premier ? Quelles règles définir pour appliquer le deuxième classifieur ? La deuxième combinaison est parallèle, et elle signifie que les classifieurs ou systèmes interagissent entre eux en même temps. Et finalement la combinaison pondérée qui consiste à associer un poids de pondération à chaque classifieur. Le poids

de pondération représente un score de confiance associé à la sortie de chaque système et la décision finale est une combinaison linéaire des différents classifieurs.

Cette approche permet de garder la précision des méthodes numériques et la robustesse des méthodes symboliques. La combinaison pourrait ainsi améliorer les performances en tirant profit des avantages de chaque méthode et éventuellement leur complémentarité [Dhaoui *et al.*, 2017].

1.3 Challenges de l'analyse d'opinions en arabe

La tâche d'analyse d'opinions en arabe est difficile. Cette difficulté provient d'une part de la langue arabe et d'autre part de la tâche elle-même. Les défis liés à la langue arabe ont été détaillés dans la section 2.5. Les challenge liés à la tâche sont liés fortement à la qualité et la quantité des ressources disponibles. Nous résumons dans la suite les difficultés d'AOA.

1.3.1 Manque de ressources

Malgré l'abondance du contenu arabe en ligne, les jeux de données et les lexiques polarisés arabes font défaut. Au cours de la dernière décennie, certains jeux de données ont été construits pour l'ASM et l'AD. Néanmoins, le nombre d'ensembles de données disponibles au public reste limité [Al-Kabi *et al.*, 2016; Assiri *et al.*, 2018]. De plus, la plupart de ces jeux de données ne contiennent pas assez de données, ce qui affecte l'évaluation des systèmes AOA par rapport aux modèles anglais car la performance des systèmes d'analyse d'opinions dépend de la taille des données manipulées. D'autre part, les difficultés liées à la construction et à l'annotation des lexiques polarisés ont empêché la disponibilité de lexiques arabes à grande échelle, en particulier avec l'existence de différents dialectes arabes et multiples domaines.

1.3.2 Opinion spam

Les opinions spams se réfèrent à des opinions fausses qui tentent délibérément d'induire en erreur des lecteurs ou des systèmes automatisés en donnant des avis positifs non méritants à certains objets cibles afin de promouvoir les objets et / ou en donnant des avis négatifs malveillants à d'autres objets afin de nuire à leur réputation.
La détection de tels spams est très importante pour les applications. L'utilité des opinions renvoie à l'utilité ou à la qualité des opinions. L'attribution automatique de valeurs d'utilité à des opinions est utile car les opinions peuvent ensuite être classées en fonction de leurs valeurs d'utilité. Avec le classement, le lecteur peut se concentrer sur ces opinions de qualité.

1.3.3 Ironie et sarcasme

Un autre challenge pour l'analyse d'opinions en arabe est l'ironie. En effet, ce phénomène linguistique est assuré par un ensemble de figures de style, telles que l'antithèse, la métaphore, l'hyperbole,

etc. En analyse d'opinions, le sarcasme ou l'ironie est traduit principalement par une opposition entre la polarité explicite de l'énoncé porté par ses mots, et la polarité implicite porté par l'énoncé lui même. Par exemple, بعد ساعتين من الانتظار، نفذت كل التذاكر، كم انا محظوظ (après deux heures d'attente, il n'y a plus de tickets, combien je suis chanceux) contient le mot محظوظ (chanceux) qui indique une polarité positive alors que l'énoncé se réfère en fait à la polarité négative.

1.4 Domaines d'application de l'analyse d'opinions

Le domaine d'analyse d'opinions connait un intérêt croissant [1]. Les domaines d'application d'analyse d'opinions sont nombreux. Dans cette section, nous présentons les domaines d'application de l'analyse d'opinions suivants : domaine académique, commercial, éducatif, politique et le domaine de la santé.

- Domaine académique :

Plusieurs campagnes d'évaluation ont été organisées. Nous citons, par exemple, SemEval2014 [Nakov & Zesch, 2014], SemEval2015 [Nakov *et al.*, 2015], SemEval2016 [Bethard *et al.*, 2016], SemEval2017 [Bethard *et al.*, 2017], Deft2017 [Benamara *et al.*, 2017], SemEval2018 [Apidianaki *et al.*, 2018], *etc*.

- Domaine commercial :

Les entreprises analysent les commentaires de clients et leurs retours afin d'améliorer la qualité des produits ou modifier la stratégie du marketing. En effet, l'analyse des commentaires dans les médias sociaux est de plus en plus considérée comme un outil d'aide à la décision pour la compréhension du marché, la catégorisation de la clientèle et la gestion de produits. Le client ou le consommateur se renseigne, avant d'acheter un produit ou un service, en analysant les retours des autres utilisateurs. Cette analyse de retours permet au client d'avoir une idée générale sur l'objet en question et le comparer éventuellement avec d'autres objets disponibles sur le marché.

Ainsi, l'analyse d'opinions s'avère utile sur le plan commercial pour l'étude de marché, et donc très utilisée dans les entreprises pour adopter les bonnes stratégies dont le but ultime est d'augmenter leurs chiffres d'affaires.

- Domaine éducatif :

L'analyse de sentiments est aussi appliquée dans le domaine de l'éducation [Altrabsheh *et al.*, 2013; Munezero *et al.*, 2013; Solís-Avilés *et al.*, 2018]. En effet, plusieurs sites web et plateformes MOOC proposent des cours en ligne. L'apprentissage en ligne est de plus en plus utilisé. Les MOOC sont suivies par plusieurs apprenants du monde entier avec seulement un ordinateur et une connexion internet. Ces plateformes offrent la possibilité de suivre les cours sans contrainte de temps. Chaque apprenant suit son cours à son rythme et selon ses disponibilités. Des certifications délivrées par ces plateformes sont également fournies pour justifier et prouver la validation de modules sous certaines conditions. Au cours de leur apprentissage, les apprenants interagissent via des forums et des espaces

1. https://trends.google.com/trends/explore?date=all&q=sentiment%20analysis

de discussion pour poser ou répondre aux questions, s'exprimer et donner des avis, *etc*. Et à la fin du module d'apprentissage, les apprenants remplissent des formulaires pour donner leurs points de vue sur le cours suivi. Afin d'évaluer leurs contenus, les MOOC mesurent la satisfaction des apprenants via leurs interactions dans les forums en cours d'apprentissage et leurs évaluations du module d'apprentissage par formulaire.

- Domaine de la santé :

L'analyse d'opinions est aussi appliqué dans le domaine de la santé [Saunders *et al.*, 2018]. En effet, l'identification ou la compilation des commentaires relatifs à la santé (ou un de ses aspects : cigarettes électroniques par exemple comme alternative au tabac [Clark, 2019]) dans des certains sites web ou réseaux sociaux est utile à la fois pour les fournisseurs de soins de santé et les professionnels de la réglementation en santé publique. L'analyse de contenus relatifs à la santé permet aussi de mesurer l'impact d'une maladie sur la personne touchée par ce problème et son entourage. Des études dans ce sens ont été menées pour des maladies chroniques [Song, 2019] comme le diabète [Gabarron *et al.*, 2019], le cancer [Modave *et al.*, 2019], *etc*.

- Domaine politique :

Les politiciens ont recours aux réseaux sociaux pour s'adresser au public et rendre leurs programmes plus accessibles aux électeurs.

L'analyse d'opinions s'applique dans plusieurs domaines. Elle intervient pratiquement afin de développer d'outils pour extraire, identifier, synthétiser et comparer des opinions.

1.5 Conclusion

Nous avons présenté, dans ce chapitre, le problème de l'analyse d'opinions. Nous avons présenté l'étymologie et la définition formelle de ce problème. Nous avons également présenté les différents types et niveaux d'analyse. Ensuite, nous avons dressé un état de l'art des différentes approches existantes (symbolique, numérique et hybride) pour la classification des opinions et leurs domaines d'application.

Cette thèse s'intéresse plus précisément à l'analyse d'opinions en arabe (AOA). Nous présentons dans le chapitre suivant la langue arabe et sa spécificité en analyse d'opinions, ainsi que l'état de l'art relatif à l'AOA.

Chapitre 2

Langue arabe et analyse d'opinions

Sommaire

2.1 Présentation de la langue arabe 22
2.1.1 Caractéristiques de l'arabe standard moderne 23
2.1.2 Caractéristiques de l'arabe dialectal 27
2.1.3 Systèmes d'écriture pour l'arabe 30
2.2 Prétraitement automatique de la langue arabe 32
2.2.1 Segmentation 33
2.2.2 Lemmatisation 33
2.2.3 Stemming 34
2.2.4 Light stemming 34
2.3 Méthodes de construction de ressources 34
2.3.1 Lexiques polarisés 34
2.3.2 Corpus 38
2.3.3 Résumé des ressources existantes 40
2.4 État de l'art de l'analyse d'opinions en arabe 42
2.4.1 Approche symbolique 43
2.4.2 Approche numérique 43
2.4.3 Approche hybride 44
2.5 Spécificité de la langue arabe pour l'analyse d'opinions 45
2.5.1 Catégories d'arabe 45
2.5.2 Dialecte arabe 46
2.5.3 Polysémie 46
2.5.4 Non voyellation 47
2.5.5 Agglutination 47
2.5.6 Entitées nommées 48
2.5.7 Négation 48

2.5.8 Système d'écriture 48
2.6 Conclusion 48

La langue arabe représente la langue officielle de 25 pays. Elle est principalement parlée par plus de 375 millions de personnes et classée la cinquième langue la plus parlée dans le monde [1]. En effet, l'arabe est originaire de l'arabe classique apparu initialement dans la péninsule arabe : c'est le berceau de la culture et de la civilisation arabe. L'arabe s'est ensuite étendu en dehors de la péninsule arabe à travers plusieurs invasions ayant pour but de répandre l'Islam.

Parallèlement, l'expansion d'internet et la prolifération de réseaux sociaux fait augmenter visiblement le nombre d'internautes dans le monde entier. L'anglais est la langue la plus utilisée sur Internet. En effet, 25% d'internautes utilisent l'anglais sur le web [2]. Le chinois et l'espagnol arrivent en deuxième et troisième places respectivement. L'arabe est classé la quatrième langue la plus utilisée par les internautes.
La croissance du nombre d'internautes est à l'origine de la recherche intense en analyse d'opinions (voir chapitre 1). Ainsi, le nombre d'internautes arabes sur internet et la forte recherche en analyse d'opinions conduisent à l'interêt considérable porté à l'analyse d'opinions en arabe ces dernières années : un intérêt principalement dû à la proportion des locuteurs arabophones au sein des médias sociaux.

Abstraction faite de tout domaine, chaque application dédiée à une langue bien déterminée prend en considération, de façon implicite ou explicite, les caractéristiques de la langue. Cela se manifeste généralement par l'utilisation de règles ou patrons sur le plan linguistique et de probabilités (ou fréquences) sur le plan statistique. Il s'en suit que la recherche en analyse d'opinions en arabe doit considérer les spécificités de la langue arabe. Ce chapitre esquisse, dans un premier temps, les principales caractéristiques de la langue arabe. Ensuite, il présente un panorama des outils TALN dédiés au prétraitement de l'arabe. Il s'intéresse, dans un deuxième temps, à la recherche en analyse d'opinions en arabe : il présente les méthodes de construction de ressources nécessaires et l'état de l'art relative à de domaine de recherche. Il dresse, enfin, les spécificités de la langue arabe pour l'analyse d'opinions.

2.1 Présentation de la langue arabe

La langue arabe représente l'une des langues les plus répandues dans le monde entier [3]. L'arabe est une langue riche et complexe au niveau de l'orthographe [Taha, 2013] et de la forme [Andrason, 2016].
Trois principales catégories ou familles d'arabe peuvent être distinguées : l'arabe classique, l'arabe

1. https://fr.wikipedia.org/wiki/Arabe
2. https://www.internetworldstats.com/stats7.htm
3. Il est difficile de déterminer précisemment le nombre de langues dans le monde. Le nombre est estimé à plus de 3 milles langues https://fr.wikipedia.org/wiki/Langue

standard moderne et l'arabe dialectal. L'arabe classique (AC) est utilisé généralement dans les écrits religieux, à savoir le Coran : le livre sacré de l'Islam. L'arabe standard moderne (ASM) est la langue officielle dans les pays arabes et principalement utilisé dans les journaux, les livres d'enseignement et les publications scientifiques. Ces deux registres classique et moderne constituent l'arabe littéral standard [Baccouche, 1974]. Quant à l'arabe dialectal (AD), il est utilisé par les locuteurs arabes dans la vie quotidienne. Ces trois variantes d'arabe peuvent coexister dans un même énoncé. Il est important de noter que l'arabe classique est utilisé surtout dans le contexte religieux, alors que l'arabe moderne et dialectal sont, de nos jours, les plus utilisés dans la vie quotidienne : on a recours à l'ASM dans des communications formelles et on utilise l'AD dans des discussions informelles. Pour cette raison, nous présentons uniquement les caractéristiques de l'arabe standard moderne et l'arabe dialectal sans détailler la complexité de l'arabe classique.

2.1.1 Caractéristiques de l'arabe standard moderne

Tout énoncé textuel est défini comme une séquence de mots formant un énoncé cohérent, porteur de sens et utilisant les règles linguistiques, syntaxiques et stylistiques propres à une langue donnée. Si on considère que la définition d'un mot, au sens graphique, est une séquence de caractères délimitée par deux séparateurs (blanc ou autre marqueur de séparation, tel que la ponctuation), alors un mot en arabe peut avoir une structure assez complexe compte tenu de l'agglutination de la langue arabe. En effet, un mot en arabe peut être formé à partir d'une base (la racine) à laquelle sont ajoutés éventuellement des affixes et/ou des clitiques. En ce qui concerne les affixes, l'arabe distingue les préfixes (agglutinés au début de la base) des suffixes (attachés à la fin de la base). De plus, il dispose d'un autre type d'affixes, dits "infixes", situés au milieu de la base. Quant aux clitiques, nous distinguons les proclitiques situés au début du mot et les enclitiques situés à la fin. [Cohen, 1970] désigne un mot graphique par "mot maximal". La figure 2.1 détaille la structure de mot graphique (mot maximal) comme proposée par [Cohen, 1970]. Un mot maximal est décomposable en : proclitique(s), forme fléchie (mot minimal) et enclitique(s). Le mot minimal constitue le noyau sémantique (l'élément porteur de sens) du mot maximal correspondant. Ce noyau est composé de : préfixe, stemme (racine) et suffixe.

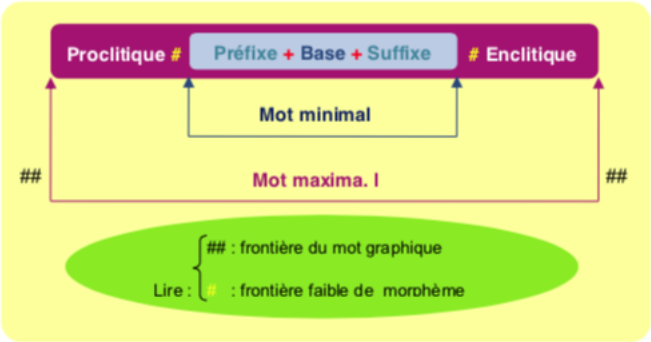


FIGURE 2.1 – Structure générale d'un mot graphique en arabe [Saâdane, 2015].

Le mécanisme d'agglutination en arabe peut générer des mots qui sont traduits en phrases complètes en français ou en anglais. Par exemple, le mot "أستحبونه" agglutine deux proclitiques (أ et سـ) et un enclitique (ـه) à la forme fléchie (تحبونـ). Cette forme agglutinée peut être traduite en "est-ce que vous allez l'aimer ?". Pour certains mots graphiques, l'agglutination peut engendrer une ambiguïté morphologique lorsqu'un clitique est assimilé à un caractère appartenant à la racine du mot. Ce phénomène s'appelle " *identité coïncidente*" [Kamir *et al.*, 2002] : c'est le fait d'obtenir un homographe similaire à un mot de base par l'ajout d'affixes ou de clitiques. C'est le cas, par exemple, de la lettre "فـ" qui fait partie de la racine "فصل" (une saison ou séparer), et qui peut être aussi considéré comme clitique au verbe وصل (lier) conjugué à l'impératif صل (lie). La voyellation peut enlever cette ambiguïté des homographes[4]. En effet, l'ajout de voyelles à ces homographes permet de distinguer فَصْلٌ (une saison) de فَصَلَ (séparer) et de فَصِلْ (et lie). Les voyelles jouent ainsi un rôle important pour lever l'ambiguïté et identifier également la catégorie grammaticale (verbe, adjectif, nom, *etc.*) de mots [Debili *et al.*, 2002]. En outre, [Attia & Somers, 2008] suppose que l'agglutination pourrait réduire cette ambiguïté avec l'hypothèse de la pyramide d'ambiguïté présentée dans figure 2.2. Elle montre que l'ajout d'affixes à la racine (ou le stemme) permet de réduire le taux d'ambiguïté. Ce taux est davantage réduit avec l'ajout de clitiques. Autrement dit, le niveau racine dans la pyramide engendre le taux d'ambiguïté le plus élevé. Pour illustrer ce constat, prenons par exemple le mot فصل, l'agglutination de l'article défini الـ comme proclitique permet d'avoir une seule possibilité الفصل (la saison). De même, l'ajout du préfixe يـ comme proclitique permet d'avoir une seule interprétation يفصل (il sépare). Similairement aux clitiques, les préfixes et les suffixes participent également au phénomène d'identité coïncidente. En effet, ils peuvent produire des formes homographiques de sens différents. Prenons à titre d'exemples la forme non voyellé أسد. Elle peut être voyellée de deux façons différentes : أَسَدٌ (un lion) et أَسُدُّ (je bloque). La première interprétation considère la forme comme un seul mot et possède une catégorie grammaticale de type nom. La deuxième interprétation est un verbe et se compose du préfixe أَ et de la racine verbale سَدَّ. Les deux interprétations coïncident au niveau de la forme surfacique non voyellée, mais elles ont deux compositions distinctes, et donc deux significations différentes.

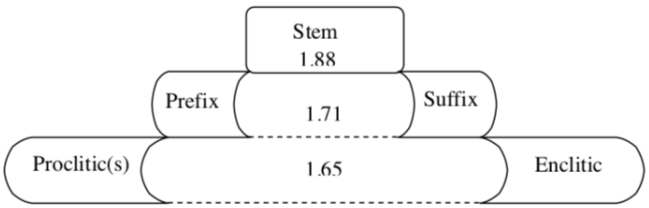


FIGURE 2.2 – Hypothèse de la pyramide d'ambiguïté [Attia & Somers, 2008].

Le phénomène d'agglutination est principalement assuré, comme expliqué ci-dessus, par des clitiques. Ces termes d'agglutination peuvent être attachés à un mot minimal pour produire une forme

4. Des homographes se sont des mots ayant la même forme othographique mais des prononciations différentes

agglutinée (un mot maximal ou un mot graphique). Le tableau 2.1 rapporte la liste des clitiques tout en distinguant les deux classes *proclitiques* et *enclitiques* [Alotaiby *et al.*, 2010].

Clitiques		Translitération	Catégorie grammaticale
Proclitiques	الـ	/Al/	Article défini
	و	/w/	Conjonction de coordination
	ف	/f/	Conjonction de subordination
	لـ	/l/	Préposition
	بـ	/b/	Préposition
	كـ	/k/	Préposition
	سـ	/s/	Particule marqueur de futur
	أ	/a'/	Article d'interrogation
Enclitiques	ي	/y/	Pronom possessif de la 1ère personne de singulier
	ـني	/ny/	pronom relatif de la 1ère personne de singulier
	ك	/k/	pronom relatif (ou possessif) de la 2ème personne de singulier
	كما	/kmA/	pronom relatif (ou possessif) du duel
	هم	/hm/	pronom relatif (ou possessif) de la 3ème personne de pluriel masculin
	هن	/hn/	pronom relatif (ou possessif) de la 3ème personne de pluriel féminin
	كم	/km/	pronom relatif (ou possessif) de la 2ème personne de pluriel masculin
	كن	/kn/	pronom relatif (ou possessif) de la 2ème personne de pluriel féminin
	نا	/nA/	pronom relatif (ou possessif) de la 1ère personne de pluriel

TABLE 2.1 – Liste de clitiques (proclitiques et enclitiques) en arabe.

Nous détaillons, dans la suite, les proclitiques et les enclitiques.

- Proclitiques :

Les proclitiques sont agglutinés au début de mot minimal. Ils sont en nombre fini (voir tableau 2.1) et sont agglutinés à des noms, des adjectifs et des verbes selon les règles linguistiques suivantes :

— Des proclitiques réservés aux noms et adjectifs : الـ, بـ et كـ.

— Des proclitiques réservés aux verbes : سـ.

— Des proclitiques généraux (agglutinés aux noms, adjectifs et verbes) : فـ, و, أ et لـ

Ce multi-usage de proclitiques avec les verbes, les noms et les adjectifs signifie implicitement que certains proclitiques peuvent jouer plusieurs rôles. En l'occurence, le proclitique و est utilisé généralement comme conjonction de coordination et de subordination, mais il peur être utilisé également comme particule d'accompagnement (واو المعية) ou de serment (واو القسم). Nous citons aussi le proclitique لـ qui peut être utilisé comme préposition (Phrase 1) ou marqueur de corroboration et renforcement de l'action *tawkïd* (Phrase 2).

Phrase 1 :
سأذهب الى المغازة لـشراء بعض الاغراض
/sA*hb AlY Almgzt l*rA'bEDAlAgrAD*/
Je vais au magasin pour faire des courses.

Phrase 2 :
لـترونّ الجحيم (le verset numéro 6 de sourate التكاثر "*le cumul des richesses*")
/lyrwn AljHym/
Vous verriez alors apparaître la Fournaise

Il est important de noter qu'une combinaison de proclitiques simples (voir tableau 2.1) est possible pour former un proclitique composé. En effet, plusieurs proclitiques peuvent être collés au mot minimal. Cette juxtaposition de proclitiques doit respecter un ordre bien défini selon la position d'apparition dans le mot maximal [Dichy, 1990]. Le tableau 2.2 présente les différentes positions possibles de proclitiques simples selon leurs apparitions dans un proclitique composé. Il est important de noter que la particule d'interrogation أ occupe toujours la première position dans un mot graphique et il est impossible qu'il soit précédé par un autre proclitique simple.

1ère position	2ème position	3ème position	4ème position
أ	و , ف	بـ, لـ, سـ, كـ	الـ

TABLE 2.2 – Agencements possibles des proclitiques.

Quatre positions sont envisageables selon des règles de compatibilité [Dichy, 1997] : chaque position est occupée par au plus un seul proclitique simple. Les règles de compatibilité exigent un agencement de proclitiques compatibles, c'est-à-dire que certains proclitiques ne sont pas compatibles entre eux et ne peuvent pas être juxtaposés. Par exemple, les particules لـ et بـ ne peuvent pas se combiner. Ainsi, l'agencement de proclitiques n'est pas aléatoire.

- Enclitiques :
Les enclitiques sont agglutinés à la fin de mot minimal. Ils sont en nombre fini (voir tableau 2.1). Nous distinguons principalement :

— les prénoms possessifs (ou génitifs) agglutinés aux noms
— les pronoms objet (ou relatifs) agglutinés aux verbes

Dans le cas échéant, s'il est conjugué à la voie passive, le verbe ne prend pas d'enclitique. Contrairement aux proclitiques, un mot graphique ne peut avoir qu'un seul enclitique.

Un mot graphique en arabe ne peut ainsi contenir qu'au plus un seul enclitique et plusieurs proclitiques. Dans la suite, nous désignons un mot graphique ou maximal par le terme "mot" tout court.

2.1.2 Caractéristiques de l'arabe dialectal

2.1.2.1 Origine

L'arabe dialectal (AD) représente une autre forme de la langue arabe. L'AD, appelé العامية (langue commune) ou الدارجة (langue courante), est utilisé dans les communications quotidiennes. Il s'agit tout simplement de l'arabe parlé [Salib, 1981]. L'AD est défini comme étant "*la langue courante des activités quotidiennes, elle est généralement parlée, bien qu'elle soit parfois écrite. Elle varie non seulement d'un territoire arabe à un autre, mais aussi d'une région à une autre au sein du même territoire*" par [Erwin, 1973] .
L'AC et l'ASM ont été considérés comme des langues écrites alors que les dialectes de l'arabe sont des langues vernaculaires (parlées). [Habash, 2010] regroupe les dialectes arabes en sept classes : arabe égyptien (EGY), arabe levantin (LEV), arabe du golfe (GLF), arabe maghrébin (MAG), arabe iraquien (IRQ), arabe yéménite (YEM) et arabe maltais (MLT).

Mot	Translitération	Traduction	Origine
فكرون	/fkrwn/	tortue	berbère
شلاغم	/*lAgm*/	moustaches	
فرملي	/frmly/	infirmier	français
تليفون	/tlyfwn/	téléphone	
سكارجي	/skArjy/	ivrogne	turc
تقاشير	/tq*yr*/	chaussettes	

TABLE 2.3 – Origine et sens de quelques mots empruntés utilisés dans l'AD.

L'AD est une variante linguistique de l'arabe standard ayant des aspects propres à chaque pays ou région. Ces aspects dévoilent la conséquence des influences linguistiques venues d'ailleurs comme le berbère, le turc, le français, l'italien, l'espagnol ou l'anglais. L'AD peut être ainsi considéré comme une mosaïque multiculturelle et un brassage de civilisations, assuré par l'emprunt de mots d'autres langues. L'emprunt représente, d'un point de vue qualitatif et quantitatif, un mécanisme dynamique assez intéressant. Il reflète l'influence des autres langues sur les dialectes locaux. Nous rapportons, dans le tableau 2.3, quelques exemples d'emprunts de mots, de différentes origines (berbère, turc et français), dans l'AD. L'emprunt se manifeste, selon [Mejri *et al.*, 2009], en trois points :

— Introduction de nouveaux suffixes empruntés
— Intégration systématique des unités empruntées dans les paradigmes construits par schèmes
— Impact phonologique qui agit par le bais de l'emprunt sur le système phonologique du dialecte

Il existe historiquement beaucoup de théories sur l'origine des dialectes arabes actuels qui remonte jusqu'aux premières langues arabes parlées ou écrites [Farghaly, 2010]. Par exemple, les linguistes [Kees, 1997] supposent que les dialectes arabes sont issus d'un premier arabe dialectal parlé pendant les premiers jours des conquêtes arabes. La conquête islamique a largement étendu l'arabe sur un vaste territoire où diverses langues étaient parlées. Les habitants des terres conquises ont parfois

adopté la langue des conquérants pour des raisons économique, sociale ou politique. Cette situation conduisait aux émergence et brassage des dialectes.
Pour aller plus loin, [Baccouche, 1974] distingue deux niveaux dans les registres de l'arabe dialectal :

— le dialectal populaire (AD familier) qui véhicule les besoins quotidiens.
— le dialectal intellectualisé qu'on retrouve dans les conversations et les émissions radiophoniques et télévisées [Boukadida, 2008]. Ce dialecte se présente comme un mélange entre l'ASM et l'AD.

2.1.2.2 Normalisation d'écriture

L'usage du dialecte dans les réseaux sociaux sur Internet participe également à la modification de l'aspect oral de l'AD, en passant d'une communication purement orale à une communication écrite, sans standard d'orthographe ni normalisation bien établis. L'absence de convention orthographique standardisée représente une difficulté énorme dans le traitement automatique de l'AD. Il fait récemment l'objet de plusieurs travaux de recherche, proposant ainsi une convention de transcription nommée CODA "*Conventional Orthography for Dialectal Arabic*". Cette convention a été proposée, dans un premier temps, pour le dialecte égyptien [Habash *et al.*, 2012], puis étendue à d'autres dialectes comme le tunisien [Zribi *et al.*, 2014], l'algérien [Saadane & Habash, 2015] et le palestinien [Jarrar *et al.*, 2014].

La convention CODA ne pose aucune contrainte sur la voyellation des textes en AD, c'est-à-dire que chacun a la liberté de voyeller ou pas son texte transcrit. Usuellement, les énoncés écrits en ASM ne sont pas voyellés, ce qui ajoute de l'ambiguïté car un même mot sans voyelles peut correspondre à plusieurs mots voyellés. Cette information liée aux voyelles est aussi disponible en AD (puisque les voyelles sont prononcées en arabe parlé). Pour cette raison, et afin de garder une similarité entre l'AD et l'ASM, CODA donne la possibilité de voyeller les mots lors de la transcription manuelle de l'énoncé oral. Autrement dit, un énoncé en AD peut être voyellé ou pas, et donc il peut être ambigu avec l'absence de voyelles comme en ASM.
En complément des ambiguïtés supplémentaires dues à la voyellation, le phénomène d'agglutination, caractéristique de l'ASM, est également présent dans l'AD. Prenons l'exemple du mot "وشريتوهشي", correspond en français à la phrase : "et est-ce que vous l'avez acheté". Cet exemple montre des ambiguïtés morphologiques. Il est composé de la particule de conjonction و, du verbe شريتو, du pronom complément هـ et de la particule d'interrogation ـشي.
Il nous importe de signaler la ressemblance entre l'agglutination de l'AD et l'ASM. Dans cette perspective, [Hamdi, 2015] propose un système de conversion du dialecte tunisien vers une forme proche de l'arabe standard pour laquelle l'application des outils conçus pour ce dernier serait fiable.

2.1.2.3 Traitement automatique de l'AD

Le traitement automatique de l'arabe dialectal attire l'attention de plusieurs chercheurs depuis les années 2010. Nous distinguons principalement deux approches de traitement. La première approche

consiste à convertir l'AD en ASM et utiliser les systèmes conçus principalement pour l'ASM [Hamdi *et al.*, 2013]. Elle est fondée sur les caractéristiques communes entre l'AD et l'ASM, présentées ci-dessus (à savoir l'agglutination et la voyellation) et peut être considérée comme une traduction de l'AD en ASM. Quant à la deuxième approche, elle considère l'AD comme une langue à part entière et développe des systèmes TALN dédiés [Boujelbane *et al.*, 2015]. Cette approche prend en considération les caractéristiques propres de l'AD et conçoit des systèmes qui manipulent directement du texte en AD. Elle ne nécessite donc pas une traduction vers l'ASM. Par contre, elle a recours à des conventions d'écriture.

2.1.2.4 Différences entre AD et ASM

Malgré les ressemblances, l'AD présente des différences avec l'ASM. Nous décrivons, dans la suite, ces différences selon quatre niveaux : phonologique, orthographique, morphologique et lexical.

— **Phonologie** : Certaines lettres de l'alphabet arabe admettent différentes prononciations selon les variantes dialectales. Par exemple, la consonne ق [q] dans l'ASM a beaucoup de variétés de prononciation en AD : (i) ق en dialecte algérien, marocain et tunisien, (ii) [g] dans quelques régions du Maghreb et dans certains dialectes bédouins orientaux, (iii) arrêt glottal en arabe égyptien et levantin. Une autre consonne en AD a également des prononciations spéciales ; il s'agit de la consonne ذ [ð] qui peut être prononcée : (i) [ð] comme le mot ذهب (or), ou (ii) [z] comme le mot ذكي (intelligent) dans certaines variantes dialectales. Il y a aussi le phonème d'arrêt de la glotte, qui apparaît dans de nombreux mots en ASM par opposition aux dialectes. Nous citons, à titre d'exemple, le mot فأس (une hache) en ASM devient فاس [fas] en AD, et le mot ذءب (un loup) en ASM devient ذيب [ðib] en AD.

— **Orthographe** : contrairement à l'ASM, les dialectes arabes n'ont pas de standard orthographique. Plusieurs variations orthographiques existent dans l'écriture des mots. Ces variations sont principalement dues aux différences phonologiques entre les dialectes arabes. Dans certains cas, la phonologie sous-jacente aboutit à une écriture d'assimilation phonologique régulière. Pour remédier à cette absence de norme, une convention d'orthographe conventionnelle a été établie pour l'arabe dialectal. À nos connaissances, CODA a été conçue uniquement pour les dialectes égyptien, tunisien, algérien et palestinien. Il faut plus d'effort pour établir des conventions pour les autres dialectes arabes ou éventuellement, mettre en place un mécanisme de portabilité entre les dialectes proches en se basant sur la classification de [Habash, 2010].

— **Morphologie** : il existe de nombreuses différences morphologiques entre l'AD et l'ASM. La première différence est la particule de futur qui apparaît sous la forme de سـ ou سوف en ASM, peut être exprimée en : (i) ح ou رح en dialecte levantin, (ii) ك en dialecte marocain, et (iii) ح ou باش en dialecte tunisien. Nous notons également que de nombreux dialectes ajoutent de nouveaux clitiques qui n'existent pas dans l'ASM, tels que l'enclitique ـش du terme de

négation ما + ـش dont l'équivalent en ASM peut être : ما, لم et لن . Par exemple, la phrase ما كتبتش القصة traduit en français "je n'ai pas écrit l'histoire".

— **Lexique** : au niveau du vocabulaire, nous pouvons trouver un nombre important de différences entre l'AD et l'ASM. Nous citons principalement les deux aspects lexicaux caractérisant les dialectes à savoir la dérivation et les emprunts [Saadane & Habash, 2015].

D'un côté, la régularité de la dérivation constitue la colonne vertébrale de la morphologie en AD. Selon [Mejri *et al.*, 2009], la dérivation en AD est enrichie par une présence relativement importante d'affixes (qui sont aussi importants pour la dérivation en ASM). Cet enrichissement est matérialisé [Inès, 2005; Sfar, 2006], à titre indicatif, par l'ajout d'un certain nombre d'affixes spécifiques comme جي qui indique la profession [Baccouche, 1994] (par exemple, كوارجي pour *un joueur de football*, قهواجي pour *un gérant d'un café*, et بنكاجي pour *un agent de banque*). Cette dérivation est spécifique : elle combine des schèmes et des affixes pour exprimer une profession.

De l'autre côté, les emprunts reflètent les interactions et les influences entre différentes langues telles que le turc, l'italien, l'espagnole, le français, l'anglais, *etc*. (voir le tableau 2.3). De plus, un mot en ASM peut être exprimé par plusieurs mots en AD. Par exemple, le mot الآن (maintenant) en ASM peut être exprimé par توا, دكا, دابا, دلوقتي dans les dialectes tunisien, marocain, algérien et égyptien respectivement.

Les dialectes arabes partagent donc des similarités avec l'ASM, à savoir la richesse morphologique et l'agglutination au sens large. Cependant, ils diffèrent grandement de l'ASM aux niveaux phonologique, orthographique, morphologique et lexical comme mentionné auparavant. Nous signalons également des différences linguistiques. Certaines d'entre elles n'apparaissent pas sous la forme écrite. Prenons par exemple, le mot لم qui est un terme de négation en ASM, mais qui signifie également "ranger" en dialecte tunisien. D'autres différences se manifestent textuellement au niveau morphologique et grammatical. La morphologie de l'ASM est plus riche que celle des dialectes arabes puisque l'AD ne dispose pas de la flexion de cas et de modes. Notamment, l'ASM a une forme duale en plus des formes singulières et plurielles, alors que dans les dialectes arabes la forme duale est absente dans la plupart des cas. D'autre part, l'AD dispose d'un ensemble de clitiques plus complexe que celui de l'ASM. Au niveau de la grammaire, l'ASM dispose d'un système de cas très complexe qui n'est pas présent dans les dialectes [Saâdane, 2015].

2.1.3 Systèmes d'écriture pour l'arabe

Globalement, tous les pays arabes ont enregistré une forte augmentation du nombre d'internautes au cours des dernières années. Le contenu arabe public sur les média sociaux se présente de deux façons : soit avec les caractères arabes ; soit avec les caractères latins. Nous présentons, dans la suite, ces deux systèmes d'écriture.

2.1.3.1 Système à base de caractères arabes

Le système d'écriture arabe, appelé alphabet *abjad*, se compose de 28 lettres regroupant 25 consonnes et 3 voyelles longues. La forme des lettres change selon la position qu'elles occupent dans le mot (au début, au milieu ou à la fin). L'arabe dispose également d'un ensemble de signes diacritiques (*harakat*) donnant origine à plus de 90 caractères [Tayli & Al-Salamah, 1990]. Ces derniers représentent les différentes combinaisons possibles entre les 28 lettres et les diacritiques. Les huit principaux signes diacritiques [Aabed *et al.*, 2007] sont catégorisés en 3 groupes [Zitouni *et al.*, 2006] comme décrit dans le tableau 2.4. Le premier groupe forme la liste des voyelles courtes : ce sont des symboles écrits soit au dessus soit au dessous de la lettre. Les diacritiques de type *tanween* représentent le deuxième groupe. Ils sont utilisés uniquement à la fin d'un mot et placés au dessus (ou au dessous) de la dernière lettre. Le troisième groupe forme les marqueurs de syllabation relatifs à la gémination (le symbole *shadda*) et la non-voyellation (le symbole *sukuun*). La plupart des textes écrits en arabe sont dépourvus de signes diacritiques. Principalement, les livres d'enseignement primaire sont voyellés comme ils sont dédiés aux élèves pour apprendre facilement la langue arabe.

Catégorie	Nom	Caractère	Prononciation
Voyelles courtes	fatha	ـَ	/a/
	damma	ـُ	/u/
	kasra	ـِ	/i/
Tanween	tanween al-fatha	ـً	/an/
	tanween al-damma	ـٌ	/un/
	tanween al-kasra	ـٍ	/in/
Marqueurs de syllabation	marqueur de gémination	xx	doublement de consonne
	marqueur de non-voyellation	ـْ	absence de son

TABLE 2.4 – Les signes diacritiques en arabe.

L'arabe est dépourvue de la notion de lettres majuscules et minuscules, elle est dite *monocamérale*. Il s'écrit de droite à gauche. Tout texte en arabe est composé d'un ensemble de phrases liées entre elles par des signes de ponctuation et des conjonctions de coordination. Nous distinguons deux types de phrases : nominale et verbale. La phrase verbale contient obligatoirement un verbe et sa structure est composée de [verbe]+[sujet]+[complément]. Par contre, la phrase nominale ne contient pas de verbe, et sa structure se compose en [sujet]+[attribut]. Il est important de noter que l'ordre des mots dans une phrase en arabe est flexible, c'est-à-dire que l'ordre de mots est relativement libre dans une phrase donnée. Cette flexibilité d'ordre de mots est très utile pour préciser l'élément sur lequel on fait le focus. En effet, si le verbe est placé au début de la phrase verbale, alors l'attention est porté sur l'action elle-même, et si c'est le sujet alors l'accent est mis sur le sujet.

2.1.3.2 Système à base de caractères latins : Arabizi

Le terme *Arabizi*, construit à partir des deux termes *Arabic* et *easy*, désigne non seulement une nouvelle manière d'écrire l'arabe mais également une évolution considérable de cette langue.
La naissance de l'Arabizi remonte aux premiers claviers d'ordinateurs et de téléphones portables, qui ne permettaient pas la saisie des caractères arabes. La première apparition de cette forme de notation était dans les SMS, le chat et les courriels informels [Elmahdy *et al.*, 2012].
Les messages rédigés en Arabizi sont écrits en caractères latins, et les lettres arabes n'ayant pas d'équivalent en latin ont été remplacées par des chiffres qui rappellent leurs formes (voir le tableau 2.5). La distinction entre voyelles courtes et longues disparaît. L'arabizi a non seulement permis l'usage d'un alphabet différent, mais il a aussi instauré l'usage de l'arabe dialectal, langue de communication familière par excellence, sous forme écrite. L'usage intensif des réseaux sociaux a aussi contribué à briser définitivement l'obstacle qui séparait l'arabe dialectal de la forme écrite.

Caractère arabe	Chiffre
ع	3
ح	7
ق	9

TABLE 2.5 – Exemples de caractères arabes et leurs équivalents en Arabizi.

L'Arabizi est fortement présent sur le web. En effet, les nouvelles technologies de communication ont influencé les dialectes qui sont passés du parlé à l'écrit mettant à la disposition des chercheurs des supports matériels pour traiter l'AD. Les locuteurs arabes utilisent le dialecte pour les échanges sur les forums, les SMS et le chat. Ces communications sont formulées soit en caractères arabes, ou aussi en caractères latins (arabe translittéré), selon le type de clavier arabe ou latin. Même s'il est écrit, le dialecte reste de l'arabe informel.

2.2 Prétraitement automatique de la langue arabe

Le traitement automatique des langues naturelles (TALN) représente une branche à part entière dans la recherche informatique. Il consiste à développer des outils ou systèmes qui permettent de manipuler les énoncés écrits. Ces outils peuvent être liés à la langue elle-même permettant de prétraiter ses spécificités indépendamment de toute tâche (par exemple, la lemmatisation, le stemming, *etc.*), ou dédiés à des tâches bien définies (à savoir, le résumé automatique, la résolution d'anaphore, analyse d'opinions, *etc.*).
La recherche en arabe dialectal est encore dans ses débuts. Compte tenu de la complexité de l'AD et le nombre de ses variantes, il n'existe pas de systèmes performants pour le prétraitement de l'AD [Alshargi *et al.*, 2019]. Pour cette raison, nous présentons uniquement les outils TALN dédiés à l'ASM. Ces derniers permettent le prétraitement de l'ASM indépendamment de toute tâche. Il regroupent

les opérations de segmentation, lemmatisation, stemming et light stemming. Nous détaillons, dans la suite, ces opérations.

2.2.1 Segmentation

L'opération qui consiste à séparer les clitiques de la forme fléchie est généralement appelée *segmentation* ou *tokenisation* [Habash & Rambow, 2005]. Celle-ci pose des problèmes d'ambiguïté dans une perspective de traitement automatique. En effet, dans certains cas, plusieurs segmentations sont possibles. Par exemple, le mot فصل admet deux segmentations possibles : (i) un seul segment فصل (une saison ou séparer) ou, (ii) deux segments ف+صل (et lie). L'ambiguïté est plus importante lorsque les diacritiques sont omis, comme c'est généralement le cas dans les énoncés arabes.

De plus, la structure des mots arabes est très complexe. En effet, si l'on considère un mot comme une séquence de caractères délimités par des séparateurs (blanc ou tout signe de ponctuation), ce mot est composé de forme fléchie et il peut contenir zéro ou plusieurs clitiques. Il peut être décomposé en proclitique(s) à son début, sous forme fléchie et enclitique(s) à sa fin. Par exemple, le mot اسيعجبه /AsyEjbh/ (va-t-il l'aimer ?) est constitué de la particule d'intérrogation ا /A/ et la particule de futur س /s/, la forme fléchie يعجب /yEjb/ et le pronom relatif ه /h/, qui sont tous agglutinés. La complexité de l'arabe est que ces clitiques se retrouvent également dans des formes fléchies. Ce dernier peut être retrouvé seul, constituant ainsi des mots. Par exemple, la particule س fait partie du verbe سمح /smh/ (permettre). Le tableau 2.6 montre la segmentation de quelques mots arabes et prouve la réduction en vocabulaire avec la segmentation. Et effet, les trois mots اسيحمله, فسيحملك et و يحملهم partagent la même forme fléchie.

Mot	**Traduction**	**Segmentation**		
		Proclitiques	**Forme fléchie**	**Enlitiques**
اسيحمله	est-ce qu'il va le prendre ?	س + ا	يحمل	ه
فسيحملك	il va te mener à	س + ف		ك
و يحملهم	et il les a mené	و		هم

TABLE 2.6 – Segmentation de quelques mots arabes.

2.2.2 Lemmatisation

La lemmatisation fait référence au processus consistant à relier un élément textuel donné au morphème lexical ou grammatical correspondant à une entrée de dictionnaire. La lemmatitasion revient donc à trouver la forme canonique de mots. Un lemme est parfois similaire à la racine après suppression des affixes de mots. Le tableau 5.2 montre l'impact de la lemmatisation sur le vocabulaire arabe.

Mot			Lemme
	Genre	Nombre	
كبيران	masculin	duel	كبير (grand)
كبيرون	masculin	pluriel	
كبيرات	féminin	pluriel	
اكتب	masculin et féminin	singulier	كتب (écrire)
يكتبون	masculin	pluriel	
تكتبان	masculin et féminin	duel	

TABLE 2.7 – Exemples montrant la réduction du vocabulaire de mots arabes avec la lemmatisation.

2.2.3 Stemming

Le stemming consiste à réduire chaque mot à sa racine (stem). La majorité des racines en arabe est composée de trois consonnes. En fait, le stemming consiste à regrouper des mots en se basant sur sur leur similitude lexico-sémantique. Par exemple, les mots : "كتب" (il a écrit), "كتبوا" (ils ont écrit), "سيكتب" (il écrira), " اكتبتم" (avez-vous écrit) ont le même contenu lexico-sémantique que "كتب" (il a écrit) qui mène au "concept d'écriture" [Jaafar *et al.*, 2017]. Ainsi, au lieu de traiter les quatre mots séparément, il est possible de traiter un seul mot après avoir réduit la liste de mots à leur racine.

2.2.4 Light stemming

Le *light stemming* désigne le processus de suppression d'un ensemble de préfixes et/ou de suffixes, sans traiter les infixes [Larkey *et al.*, 2007]. Par exemple, les deux light stems suivants روعة /rw't/ (une splendeur) et روّع /rw'̃/ (effrayer) ont le même stem راع /rA'/. Cependant, ils ont deux sémantiques différentes : روعة avec une polarité positive et روّع avec une polarité négative.

2.3 Méthodes de construction de ressources

Nous présentons, dans cette section, les méthodes utilisées pour construire les ressources nécessaires pour l'analyse d'opinions en arabe (AOA), à savoir lexiques polarisés et corpus. Nous rapportons également la liste de ressources disponibles.

2.3.1 Lexiques polarisés

Un lexique polarisé se compose d'un ensemble de couples (ω, s) où ω : un mot (ou un groupe de mots) et s : un score de polarité associé à ω. Les mots ω sont des mots polarisés qui sont représentés principalement par les catégories syntaxiques de type adjectif, nom et verbe (voir chapitre 1). Un lexique polarisé ne contient pas, par exemple, les *mots outils*, dit aussi *mots vides* en français et *empty words* ou *stop words* en anglais. Une liste de mots outils contient les mots ayant peu de valeur

sémantique tels que : les prépositions, les pronoms, les déterminants, *etc*.
Un score de polarité *s* est associé à chaque mot dans le lexique polarisé. Ce score, représenté par une valeur réelle, reflète le degré (ou l'intensité) de positivité ou de négativité selon une échelle à valeurs entières ou réelles. Par exemple, selon l'échelle $[-5, 5]$, le réel -5 représente une négativité absolue et le réel 5 représente une positivité absolue. Les réels inférieurs à 0 désignent différentes intensités de négativité : plus le réel est loin de 0 (et proche de -5), plus la négativité est forte (ou intense). Et les réels supérieurs à 0 représentent différentes intensités de positivité : la positivité est plus forte lorsque le réel est plus proche de 5. Le réel 0 signifie que le mot n'est ni positif ni négatif : il reflète une polarité neutre.

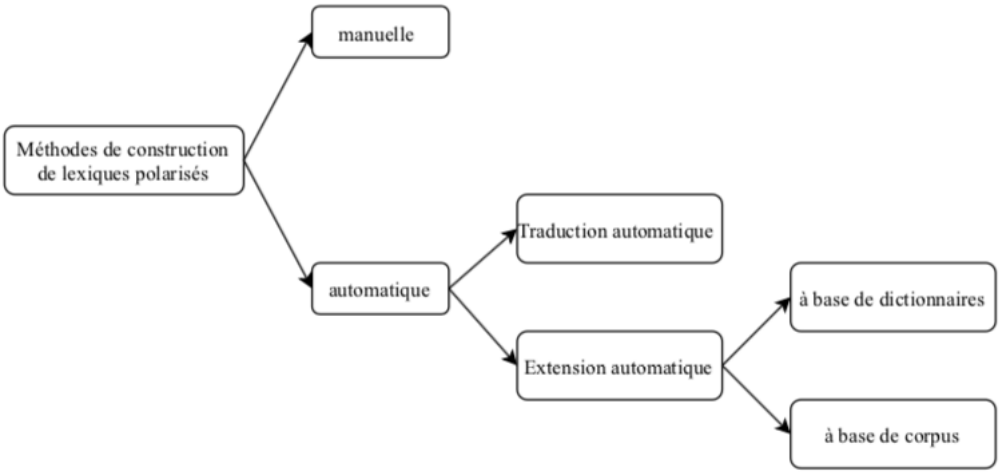


FIGURE 2.3 – Méthodes de construction de lexiques polarisés.

La construction de lexiques polarisés peut se faire deux façons différentes : manuelle ou automatique. La figure 2.3 résume les différentes techniques de construction de lexiques polarisés. Nous présentons, dans la suite, ces deux façons de construction.

a) Construction manuelle

La construction manuelle nécessite des experts. Elle consiste à déterminer un ensemble de mots polarisés et préciser leurs polarités. La polarité peut être portée par des verbes, des noms, des adjectifs, *etc*. Les premiers travaux en construction de lexiques supposaient que l'opinion était généralement exprimée via des adjectifs et s'intéressaient uniquement à la polarité des adjectifs [Abdul-Mageed & Diab, 2012a]. Pour cette raison, les premiers lexiques contiennent uniquement des adjectifs polarisés.

Les premiers lexiques en AOA ont été construits manuellement. Nous citons par exemple le lexique établi par [Abdul-Mageed *et al.*, 2011] qui regroupe 3 982 adjectifs étiquetés : positive, négative ou neutre. Les adjectifs ont été extraits d'articles de news (une partie du corpus de news ATB [Maamouri *et al.*, 2004]). Ce lexique a été raffiné par [Abdul-Mageed & Diab, 2012a] pour introduire un nouveau lexique *Sifaat*. Ce dernier contient 3 325 adjectifs en arabe étiquetés en positive, négative ou neutre. Les adjectifs ont été annotés séparément par deux annotateurs. Les cas de désaccord ont

été réglés après une discussion entre les annotateurs. En plus des polarités, *Sifaat* a été enrichi par la traduction en anglais de chaque adjectif. [Abdul-Mageed & Diab, 2012a] ont construit un autre lexique composé de 29 991 mots les plus fréquents dans un corpus de commentaires sur Youtube, et qui sont manuellement annotés en positive, négative ou neutre.

En plus de mots polarisés, des proverbes peuvent être utilisés pour exprimer une opinion. Étant une expression propre à un langage, un proverbe est composé d'une liste de mots et utilisé dans un contexte défini avec une seule signification. Il ne peut pas être compris à partir des significations individuelles de ses mots et peut donner une polarité différente quand il est traité comme des mots séparés. Dans cette perspective, [Ibrahim *et al.*, 2015b,c] ont mis en place le lexique *AIPSeLex* composé de 3 296 expressions et proverbes (annotés manuellement) utilisés dans le dialecte égyptien. Généralement, les lexiques établis de façon manuelle sont petits et ne couvrent pas tous les mots polarisés d'une langue donnée.

b) Construction automatique

La construction automatique de lexiques polarisés regroupe les techniques de : i) traduction automatique et ii) extension automatique.

- Traduction automatique :

La construction de lexiques polarisés consiste à traduire des lexiques existants en d'autres langues. La langue source la plus utilisée est principalement l'anglais vu la disponibilité et la qualité des lexiques existants en anglais. Elle se fonde sur des techniques de traduction automatique. Dans le cadre de l'AOA, plusieurs lexiques ont été construits par traduction automatique. Nous citons, par exemple, les travaux de [Mourad & Darwish, 2013; Abdul-Mageed & Diab, 2014; Abdulla *et al.*, 2014a,b; Salameh *et al.*, 2015; Al-Twairesh *et al.*, 2016].

- Extension automatique :

La construction de lexiques polarisés nécessite un jeu initial de mots polarisés, généralement établi de façon manuelle. Elle étend automatiquement ce jeu de mots en lui ajoutant un nouvel ensemble de mots polarisés. Plusieurs techniques d'expansion ont été proposées pour générer de tels lexiques. Ces techniques peuvent être classées en deux familles : la première est basée sur des dictionnaires et la deuxième est fondée sur des corpus.

Les techniques à base de dictionnaires utilisent des dictionnaires de synonymes et antonymes ou des ontologies. Sur le plan de dictionnaires, pour chaque mot polarisé $w\omega$ du jeu initial, le nouvel ensemble est enrichi par les synonymes de ω en leurs attribuant la même polarité de ω, et par les antonymes de ω en leurs associant l'opposée de la polarité de ω. Nous citons, par exemple, le lexique *ArSeLex* [Ibrahim *et al.*, 2015c], qui regroupe 5 244 mots (2 003 positifs, 2 829 négatifs et 412 neutres), construit par extension d'une liste de 400 adjectifs avec des synonymes et des antonymes. [Mobarz *et al.*, 2014] ont exploité les relations de synonymie, antonymie et causalité pour construire leur lexique polarisé *SentiRDI*. Nous citons également les travaux de [Abdulla *et al.*, 2013; Al-Rowaily *et al.*, 2015; Alhumoud *et al.*, 2015a,b].

Sur le plan des ontologies, pour chaque mot polarisé ω du jeu initial, le processus consiste à associer

à chaque élément du synset de ω dans l'ontologie un score de positivité, de négativité et d'objectivité. Par exemple, [Mahyoub *et al.*, 2014] ont utilisé l'ontologie *Arabic WordNet* pour associer des scores de subjectivité aux mots qui le constituent, formant ainsi un lexique de 7 576 mots (885 positifs, 616 négatifs et 6 075 neutres). [Aldayel & Azmi, 2016] ont utilisé l'ontologie *SentiWordNet* pour extraire un lexique de 1 500 mots (500 positifs et 1 000 négatifs). [Abuaiadh, 2013] ont également utilisé *SentiWordNet* et l'analyseur morphologique *AraMorph* pour construire le lexique *SLSA* reliant les glosses de *AraMorph* aux synsets de *SentiWordNet*.

En ce qui concerne la deuxième famille de techniques utilisées pour l'extension automatique de lexiques, ces techniques se basent sur des corpus annotés. C'est-à-dire, pour chaque mot ω du jeu initial, de nouveaux mots polarisés sont extraits à partir d'un corpus donné par calcul de la fréquence et la co-occurence de mots. Par exemple, [Abdulla *et al.*, 2013] ont extrait une liste de mots positifs et négatifs avec une pondération de fréquence de mots (TF) (*term frequency*) d'un corpus de commentaires équilibré (positifs et négatifs). [ElSahar & El-Beltagy, 2015] ont extrait des descripteurs de type unigramme et bigramme à partir d'un corpus de commentaires et ils ont utilisé l'algorithme SVM pour sélectionner les descripteurs les plus significatifs.
En plus de la fréquence de termes, l'information de co-occurence de mots peut être utilisée pour étendre un jeu initial de mots polarisés. Une des méthodes les plus répandues est basée sur le calcul d'information mutuelle *Pointwise Mutual Information* (PMI). L'équation 2.1 détaille le calcul de la PMI entre le mot candidat ω et la classe c, avec c $\in$ {positive, négative}. Le score de polarité (SP) associé à ω représente la différence entre PMI(ω, positive) et PMI(ω, négative) (voir équation 2.2) : si la différence est positive, alors la classe positive est attribuée au mot ω avec le score SP ; et dans le cas contraire, la classe négative est associée à ω.

$$PMI(w,c) = log\frac{p(c|w)}{p(c)} \tag{2.1}$$

$$SP(w) = \text{PMI(w, positive)} - \text{PMI(w, négative)} \tag{2.2}$$

[Al-Twairesh *et al.*, 2016] ont utilisé la méthode basée sur la PMI pour construire le lexique *AraSenTi*. Nous citons également les travaux de [El-Beltagy & Ali, 2013; Abdul-Mageed & Diab, 2014; ElSahar & El-Beltagy, 2014].

La qualité du lexique polarisé intervient dans les performances de systèmes de détection de polarité. Elle dépend fortement des polarités attribués, manuellement ou automatiquement, aux mots qui composent le lexique. L'attribution de polarité à un mot demeure une tâche difficile. En effet, la polarité d'un mot dépend du domaine. Par exemple, le mot كبير/kbyr/ (grand) admet une polarité positive si le domaine est la taille d'une chambre dans un hôtel, et une polarité négative si le domaine est la taille d'une batterie de téléphone ou de caméra. De plus, la polarité d'un mot change selon la culture [Duwairi, 2015; Saif *et al.*, 2016]. La langue et la manière dont elle est exercée sont très sensibles à la culture.

La méthode manuelle pour la création de lexiques est plus précise que la méthode automatique, mais elle nécessite plus de temps et de travail [Abdulla *et al.*, 2014a], et selon l'étude [Abdulla *et al.*, 2013], plus la taille du lexique est grande plus la précision du classifieur qui utilise le lexique est élevée. Afin de tirer profit de leurs avantages, plusieurs travaux ont combiné les méthodes de construction de lexique [Badaro *et al.*, 2014a, 2018; Youssef & El-Beltagy, 2018].

2.3.2 Corpus

Les corpus représentent une ressource importante pour l'entrainement et l'évaluation des systèmes d'AO. La plupart des corpus pour l'analyse des sentiments sont collectés sur des sites Web et des plateformes de médias sociaux. Les méthodologies de collecte les plus utilisées sont l'exploration du Web *web crawling* et l'appel des interfaces de programmation d'application Web, désignée par API pour *Application Programming Interface*, telles que l'API de Twitter et l'API de Facebook.
Deux familles de construction de corpus peuvent être distinguées : manuelle et automatique. Cette catégorisation se base sur le type d'annotation. Nous détaillons, dans la suite, ces deux familles de construction de corpus.

a) Construction manuelle :
Une grande partie des corpus en analyse d'opinions en arabe a été manuellement construite, c'est-à-dire annotée par des annotateurs humains (arabes natifs dans la majorité des cas).
Le corpus *Opinion Corpus for Arabic* (OCA) [Rushdi-Saleh *et al.*, 2011b] représente le premier corpus en AOA. Il contient 500 critiques de films recueillies sur différents blogs et pages Web arabes et annotées manuellement (250 critiques positives et 250 critiques négatives). [Abdul-Mageed & Diab, 2012b] a présenté le corpus AWATIF : un corpus multi-genres comprenant 10723 phrases en arabe extraites de trois sources : l'ATB, une sélection de forums Web, et une liste de pages de discussion sur Wikipedia. Les phrases ont été annotées manuellement en trois classes : objective, positive ou négative. La même équipe [Abdul-Mageed & Diab, 2014] a présenté une combinaison de corpus (annoté en positive et négative) de différents genres : DARDASHA : une collection de 2 798 messages de discussion extraits du site web Maktoub, TAGREED : un ensemble de 3 015 tweets en arabe, TAHRIR : un ensemble composé de 3 008 phrases provenant des pages de discussion de Wikipedia, et MONTADA : un ensemble de données comprenant 3 097 phrases extraites de forums sur le web. [Rahab *et al.*, 2019] a introduit le corpus SANA : un corpus de 513 commentaires extraits de journaux annoté manuellement en positive, négative et neutre par deux locuteurs natifs. Ce corpus contient 236 commentaires positifs, 194 commentaires négatifs et 83 commentaires neutres.

Twitter constitue une source importante de construction de corpus. La longueur des messages représente une caractéristique fondamentale des documents. Elle est limité à 140 caractères au maximum dans les tweets. En effet, plusieurs travaux en AOA ont utilisé twitter comme source de construction de corpus. Nous citons, par exemple, le travail de [Mourad & Darwish, 2013] qui a introduit un

corpus de 2 300 tweets annotés manuellement en positive ou négative par des annotateurs natifs. De même, deux experts humains ont annoté le corpus équilibré *ArTwitter* [Abdulla *et al.*, 2013] composé de 1 000 tweets positifs et 1 000 tweets négatifs. Un autre corpus de tweets (SDCT) a été établi par [Al-Thubaity *et al.*, 2018] et contient 5 400 tweets en dialecte saoudien. SDCT a été annoté avec la méthode "*crowdsourcing*" qui consiste à faire l'annotation en faisant appel aux services d'un certain nombre de personnes, rémunérées ou non, généralement via Internet. L'annotation avec *crowdsourcing* a été aussi utilisée pour créer le corpus de tweets *ASTD* [Nabil *et al.*, 2015] composé de 10 000 tweets annotés en objective, positive, négative et mixte. Il existe d'autres corpus construits à partir de twitter [Ibrahim & Salim, 2013; Al-Osaimi & Badruddin, 2014; Refaee & Rieser, 2014a,b; Ibrahim *et al.*, 2015c; Assiri *et al.*, 2016].

En plus de Twitter, plusieurs travaux en AOA ont utilisé Facebook pour créer leurs corpus et évaluer leurs systèmes. Les posts sur Facebook sont caractérisés par une longueur variable allant d'un mot à un paragraphe. Dans cette perspective, nous citons le travail de [Itani *et al.*, 2017] qui a créé deux corpus : le premier regroupe 1 000 posts extraits de la page Facebook "*Al Arabiyya News*" et le deuxième contient 1 000 posts collectés de la page Facebook "*The Voice*". Ces deux corpus ont été annotés manuellement par quatre experts natifs en cinq classes : positive, négative, mixte, spam et neutre. [Medhaffar *et al.*, 2017] ont présenté le corpus *Tunisian Sentiment Analysis Corpus* (TSAC) composé de 17 060 publications sur Facebook en dialecte tunisien. Ces posts ont été annotés manuellement en positive et négative formant ainsi 8 215 commentaires positifs et 8 845 commentaires négatifs. TSAC a été recueilli sur des pages Facebook officielles de radios et chaînes de télévision tunisiennes. D'autres corpus ont été construits à partir de Facebook, nous citons à titre d'exemple [Hamouda & El-taher, 2013; Soliman *et al.*, 2014; Badache *et al.*, 2019; Maghfour & Elouardighi, 2018; Badache, 2019].

La tâche d'annotation manuelle de corpus est coûteuse en terme de temps et nécessite des ressources humaines (experts ou locuteurs natifs). Elle est fortement affectée par la subjectivité des annotateurs [Bosco *et al.*, 2013]. Par conséquent, il est recommandé d'utiliser au moins deux annotateurs et de mesurer l'accord inter-annotateur. Le coefficient kappa de Cohen [Artstein & Poesio, 2008] est la mesure couramment utilisée pour déterminer l'accord.

b) Construction automatique :

La construction automatique est basée sur les évaluations des commentaires. En effet, certains sites web offrent la possibilité de commenter via le texte et le nombre d'étoiles. Autrement dit, l'internaute exprime son opinion, à la fois, via le contenu textuel de son commentaire, et aussi à travers le nombre d'étoiles qu'il attribue à son commentaire. Le système d'attribution d'étoiles, dit aussi *star rating system*, est très répandu principalement dans les sites de e-commerce tels que Amazon, Ebay, *etc*. Le nombre d'étoiles attribué à chaque commentaire représente l'évaluation de l'entité en question par le détenteur du commentaire. Généralement, une échelle de 1 à 5 étoiles est utilisée dans la majorité des sites web. Ce nombre d'étoiles reflète le degré ou l'intensité de positivité ou de négativité de l'opinion. Il est d'usage, dans la communauté scientifique, de considérer les commentaires avec 4

ou 5 étoiles comme des commentaires positifs, tandis que les commentaires avec 1 ou 2 étoiles sont considérés comme commentaires négatifs. Les commentaires avec 3 étoiles sont considérés comme des commentaires neutres (voir tableau 2.8).

Nombre d'étoiles	Polarité
⋆⋆⋆⋆⋆	positive
⋆⋆⋆⋆	
⋆⋆⋆	neutre
⋆⋆	négative
⋆	

TABLE 2.8 – Correspondance entre le nombre d'étoiles et la polarité.

Le système d'attribution d'étoiles suppose une correspondance implicite entre le contenu textuel et le nombre d'étoiles associé. C'est-à-dire pour un commentaire quelconque, le nombre d'étoiles devrait refléter la même polarité que celle portée par le contenu textuel. Si le contenu textuel exprime une polarité positive, alors le nombre d'étoiles doit aussi refléter la polarité positive (4 ou 5 étoiles), et inversement. Cette correspondance de polarité entre le texte et le nombre d'étoiles masque une difficulté de traitement des commentaires annotés avec 3 étoiles. En effet, l'attribution de 4 (respectivement 2) étoiles signifie que le texte pourrait contenir des parties négatives (respectivement positives), et la polarité du texte en entier est positive (respectivement négative). L'attribution de 3 étoiles demeure ambigue : l'internaute pourrait utiliser 3 étoiles pour exprimer une opinion neutre, ou une opinion positive (il considère, dans ce cas, 3 étoiles comme la borne inférieure de positivité), ou une opinion négative (3 étoiles est considéré comme la borne supérieure de négativité). Nous discutons davantage, dans le chapitre 4, la construction automatique de corpus basée sur le nombre d'étoiles.

2.3.3 Résumé des ressources existantes

De nombreux travaux ont été récemment réalisés afin de mettre à la disposition de la communauté TALN des ressources et améliorer les systèmes d'AOA. Nous présentons, dans cette section, les ressources d'AOA les plus citées. Nous nous sommes également basés sur le critère de disponibilité pour choisir nos ressources.

2.3.3.1 Corpus

Le tableau 2.9 montre les corpus les plus utilisés. Nous évaluons chaque corpus en se basant sur les critères suivants : niveau d'analyse, taille, domaine, source, construction et Arabizi. Les corpus sont triés par taille pour les différents niveaux d'analyse. La colonne *construction* reflète la qualité du corpus et la fiabilité de l'annotation. La construction manuelle de corpus signifie que la ressource a été annotée par des annotateurs humains, alors que la construction automatique de ressources signifie que les polarités des documents ont été attribuées en se basant sur le système *star rating* sans vérification humaine.

Corpus	Niveau	Taille	Domaine	Source	Construction	Arabizi
BRAD	document	510k	livres	web	automatique	∅
HARD		373k	hôtels	web	automatique	∅
LABR		63k	livres	web	automatique	∅
TSAC		17k	multi	facebook	manuelle	✓
HTL		15k	hôtels	web	automatique	∅
Prod		15k	produits	web	automatique	∅
RES		11k	restaurants	web	automatique	∅
MOV		1,5k	films	web	automatique	∅
OCA		500	films	web	manuelle	∅
AraSenTi-Tweet	phrase	17k	-	twitter	manuelle	∅
ASTD		10k	-		manuelle	∅
ArSentD-LEV		4k	-		manuelle	∅

TABLE 2.9 – Corpus disponibles gratuitement pour l'analyse d'opinions en arabe.

Dans ce contexte, Le corpus LABR, introduit par [Aly & Atiya, 2013; Nabil *et al.*, 2014], contient 63 257 critiques de livres en ASM et AD. Les documents de LABR sont annotés en étoiles sur une échelle allant de 1 à 5 étoiles.
De façon similaire à LABR, [Elnagar & Einea, 2018; Elnagar *et al.*, 2018b] a construit le corpus BRAD qui est composé de 510 598 critiques sur des livres annotés en étoiles sur une échelle allant de 1 à 5 étoiles. Dans le domaine des hôtels, [Elnagar *et al.*, 2018a] a mis en place le corpus HARD qui est composé de 373 750 critiques sur hôtels annotés en étoiles sur une échelle allant de 1 à 5 étoiles. [ElSahar & El-Beltagy, 2015] a fourni quatre corpus : l'ensemble de données MOV contenant 1,5k critiques sur films, l'ensemble de données HTL regroupant 15k critiques sur hôtels, l'ensemble de données RES contenant 11k critiques sur restaurants et l'ensemble de données PROD composé de 15k critiques sur produits. Ces quatre corpus sont classés en positive, négative ou mixte.
[Medhaffar *et al.*, 2017] introduit le corpus TSAC contenant 17k commentaires sur facebook annotés manuellement en positive et négative. Ces commentaires sont en dialecte tunisien et sont écrit en Arabizi.
Le corpus AraSenTi-Tweet contient 17k tweets en dialecte saoudien annotés en positive, négative, mixte et neutre [Al-Twairesh *et al.*, 2017]. Le corpus ASTD se compose de 10k tweets en dialecte égyptien, classés manuellement en objectif, positif, négatif et mixte [Nabil *et al.*, 2015]. Le corpus ArSentD-LEV contient 4k tweets en AD (Jordanie, Syrie, Palastine et Liban) [Baly *et al.*, 2019].

2.3.3.2 Lexiques polarisés

Le tableau 2.10 montre les lexiques polarisés disponibles gratuitement. Les lexiques sont triés par ordre décroissant selon la taille. Le lexique *ArSenL* regroupe 4 616 mots annotés avec des étiquettes morpho-syntaxiques, des scores de positivité et de négativité, des offsets en *Arabic WordNet*,

etc [Badaro *et al.*, 2014b]. [ElSahar & El-Beltagy, 2015] a construit 5 lexiques avec différents domaines : hôtel, restaurant, film, livre et produit, dont les tailles sont respectivement 217, 733, 86, 873 et 363. [Saif M. Mohammad, 2016] a créé automatiquement des lexiques polarisés en utilisant deux méthodes différentes : (1) utiliser PMI sur des tweets arabes, et (2) traduire automatiquement en arabe des lexiques polarisés anglais. Trois lexiques sont construits sur des tweets arabes : lexique arabe d'émotion (43 308 mots), lexique de hashtags (22 006 mots) et lexique dialectal (20 127 mots dialectaux). Et quatre lexiques traduits de l'anglais vers l'arabe qui sont MPQA (8 221 mots), S140 (26 740 mots), NRC (32 582 mots) et le lexique de Bing Liu (6 789 mots). Le lexique *Arabic senti-lexicon* contient 3 880 synsets positifs et négatifs annotés avec des étiquettes morpho-syntaxiques, la polarité, les scores de positivité et de négativité, et une liste de synonymes [Al-Moslmi *et al.*, 2018]. Le lexique *SLSA* contient 35 000 mots étiquetés avec leurs étiquettes morpho-syntaxiques, leurs scores de positivité, négativité et objectivité [Eskander & Rambow, 2015].

Lexique	**Taille**	**Score de positivité**	**Score de négativité**	**Polarité**	**Arabizi**
lexique d'émotions	43k	-	-	✓	∅
SLSA	35k	✓	✓	-	∅
NRC	32k	-	-	✓	∅
S140	26k	-	-	✓	∅
lexique de hashtags	22k	-	-	✓	∅
lexique dialectal	20k	-	-	✓	∅
MPQA	8k	-	-	✓	∅
lexique de Bing Liu	6k	-	-	✓	∅
ArSenL	4k	✓	✓	-	∅
Arabic senti-lexicon	4k	✓	✓	✓	∅
lexique-livre	873	-	-	✓	∅
lexique-restaurant	733	-	-	✓	∅
lexique-produit	363	-	-	✓	∅
lexique-hôtel	217	-	-	✓	∅
lexique-film	86	-	-	✓	∅

TABLE 2.10 – Lexiques disponibles gratuitement utilisés dans cette thèse.

2.4 État de l'art de l'analyse d'opinions en arabe

Un intérêt récent en AOA porte sur l'analyse au niveau aspect. Cependant, la majorité des travaux en analyse d'opinions sont réalisés aux niveaux document et phrase. Cette section donne un aperçu sur les différents travaux réalisés en approches d'AOA : symbolique, statistique et hybride (voir chapitre 1). Nous nous intéressons principalement aux travaux effectués aux niveaux document et phrase.

2.4.1 Approche symbolique

Les travaux en approche symbolique ne sont pas nombreux. [Almas & Ahmad, 2007; Farra *et al.*, 2010] proposent une méthode s'appuyant sur une grammaire locale pour déterminer la polarité de document financier. Cette grammaire regroupe un ensemble de patrons. L'identification de tels patrons dépend fortement du style de documents. En effet, les documents financiers respectent un style d'écriture bien défini, ce qui permet d'extraire des patrons fiables pour la détermination de polarité.
Nous citons également le travail de [Abdulla *et al.*, 2014b] qui construisent manuellement un lexique contenant 4 815 mots (1 942 mots positifs et 2 873 mots négatifs) et le testent pour classer des commentaires. Leur système calcule le nombre de mots positifs et négatifs dans un texte donné afin de générer sa polarité globale.
Enfin, [Al-Kabi *et al.*, 2014] ont mis en place un outil qui détermine la subjectivité, la polarité d'une opinion et son intensité. Ils utilisent deux lexiques généraux et seize lexiques spécifiques.

2.4.2 Approche numérique

Afin d'obtenir un système fiable basé sur une approche numérique, la conception de bonnes features constitue l'étape primordiale pour la classification. Les sacs de mots et les n-grammes de caractères représentent les features les plus communes en AOA [Al-Ayyoub *et al.*, 2019].
[Abbasi *et al.*, 2008] se sont concentrés sur l'extraction et la sélection de features pour l'analyse d'opinions issues de commentaires (en anglais et en arabe) dans des forums web. Ils ont proposé des features stylistiques et syntaxiques et ont utilisé l'algorithme génétique pondéré par entropie pour la sélection de features pertinentes à l'analyse d'opinions. [Abdul-Mageed *et al.*, 2014] ont mis en place le système SAMAR pour la détection de subjectivité et de polarité dans les réseaux sociaux et testent l'impact des attributs de type étiquettes morphosyntaxiques. Ces derniers s'avèrent utiles pour la classification en subjectivité.
Généralement, les features les plus utilisées en approche numérique pour AOA peuvent être classées en trois catégories. Le tableau 2.11 résume la majorité de features utilisées en AOA.

Features	Exemples
Surfaciques	présence ou nombre d'occurences
	TF, TF-IDF
	longueur de textes
	n-grammes
	ponctuation
Syntaxiques	étiquettes morpho-syntaxiques
	arbre syntaxique
Lexicales	présence ou nombre d'occurrence ou score de mots polarisés

TABLE 2.11 – Liste de features utilisées pour AOA.

Plusieurs travaux en AOA ont été réalisés dans le cadre d'approche numérique. Par exemple, [Bayoudhi *et al.*, 2015] ont comparé trois classifieurs : SVM, NB et un réseau de neurones sur les

corpus OCA [Rushdi-Saleh *et al.*, 2011a] et ACOM [Mountassir *et al.*, 2013b].
[Hadi, 2015] a utilisé les deux méthodes SVM et NB pour classifier un ensemble de 3 700 tweets en positive, négative ou neutre. Le système SAMAR [Abdul-Mageed *et al.*, 2014] analyse la subjectivité et la polarité d'un énoncé textuel donné (écrit en ASM ou dialecte égyptien). Les auteurs ont utilisé plusieurs corpus extraits des médias sociaux, et la variante "*SVM light*" de l'agorithme SVM.
Nous citons également le travail de [Medhaffar *et al.*, 2017] qui a testé les classifieurs SVM et NB pour analyser l'opinion exprimée dans des commentaires écrits en dialecte tunisien. Dans [Itani *et al.*, 2012], les auteurs ont exploité un modèle NB pour classifier automatiquement des posts Facebook écrits en plusieurs dialectes arabes (syriens, égyptiens, irakiens et libanais).
[Guellil *et al.*, 2018b] a utilisé les représentations vectorielles discrètes de type sac de mot (BOW *Bag of words*) et les représentations vectorielles continues de type Doc2vec [Mikolov *et al.*, 2013b] afin d'extraire les features utiles pour évaluer leur corpus de 8000 posts sur Facebook (4000 posts écrits en arabe et 4000 écrits en Arabizi). Les auteurs ont testé différents algorithmes de classification tels que SVM, NB, RL et arbre de décision et ont obtenu une bonne performance avec la régression logistique avec un score F1 de 72% pour l'arabe et 78% pour l'Arabizi. Afin d'améliorer les résultats de la classification de l'Arabizi, [Guellil *et al.*, 2018a] ont proposé d'ajouter une composante de translittération convertissant l'Arabizi en arabe.
[Gamal *et al.*, 2019] ont testé différents algorithmes d'apprentissage automatique tels que SVM, NB, Ridge Regression (RR), et Adaboost sur un corpus de 151548 tweets. Pour extraire les features, les auteurs ont utilisé TF-IDF et ont obtenu leurs meilleurs résultats en utilisant RR avec 99,9% de F1.

Nous présentons les travaux utilisant les réseaux de neurones dans le chapitre 3.

2.4.3 Approche hybride

Le premier travail en approche hybride est celui de [El-Halees, 2011]. Il a proposé une hiérarchie séquentielle de classifieurs. Son approche combine trois méthodes successives : une méthode basée sur le lexique, un modèle d'entropie maximale et le modèle k-le plus proches voisins. Il a montré le gain relatif à la combinaison.

En approche hybride, la majorité des travaux fait recours aux lexiques de mots polarisés par ajout de features lexicales (voir tableau 2.11) à l'ensemble de descripteurs du classifieur. [Ibrahim *et al.*, 2015d] a utilisé un lexique de 5244 adjectifs, un lexique de 3296 idiomes pour améliorer la classification de phrases avec un SVM. [Refaee & Rieser, 2016] ont appliqué une approche hybride pour la prédiction de l'intensité de la polarité dans les tweets. Ils ont utilisé particulièrement la régression logistique pour prédire les scores initiaux qui sont ajustés en appliquant des règles extraites à partir d'un lexique polarisé.
Dans le travail de [Hedar & Doss, 2013], les auteurs ont utilisé un classificateur SVM. Pour faire cette classification, les auteurs utilisent un lexique contenant 1 300 mots dont 600 sont positifs et 700 négatifs. Ce travail s'appuie sur le dialecte égyptien. Les résultats expérimentaux ont montré que l'utilisation du lexique améliore considérablement les résultats.

[Khalifa & Omar, 2014] a proposé une méthode hybride fondée sur un lexique et un classificateur NB en même temps. La méthode proposée est précédée d'une phase de pré-traitement (normalisation, segmentation, *etc.*). Le lexique intervient pour remplacer les mots avec leurs synonymes polarisés. [Elshakankery & Ahmed, 2019] a proposé le système d'analyse hybride (HILATSA) qui combine des approches basées sur le lexique et sur l'apprentissage automatique afin d'identifier les polarités des sentiments de tweets. L'approche proposée a été testée à l'aide de différents jeux de données. Il a obtenu une exactitude de 73,67% pour le problème de classification à 3 classes et de 83,73% pour la classification binaire. La composante d'apprentissage semi-automatique s'est révélée efficace, car elle a amélioré la précision de 17,55%. Dans le même contexte, [Touahri & Mazroui, 2019] ont évoqué l'apprentissage supervisé pour la construction de leur modèle afin de classer un ensemble de critiques. Ils ont appliqué leur approche aux différents corpus proposés par [ElSahar & El-Beltagy, 2015], tels que HTL, PROD, *etc.* Les meilleurs résultats ont été obtenus en combinant différentes fonctionnalités et ils ont atteint 93,84% d'exactitude sur le corpus HTL.

2.5 Spécificité de la langue arabe pour l'analyse d'opinions

La prolifération des médias sociaux sur le web et sa richesse en expressions d'opinions et de sentiments a augmenté l'intérêt de la communauté de recherche. La masse de données en arabe est considérable compte tenu du nombre important d'internautes s'exprimant en arabe.
Malgré la croissance récente du contenu public en arabe sur le web et le développement continu des outils de TALN en arabe, la recherche en analyse d'opinions en arabe fait toujours face à des défis et des challenges. Ces derniers peuvent être classés en deux familles : les défis liés à la tâche d'analyse d'opinions (voir section 1.3 du chapitre 1 et ceux reliés à la langue arabe elle-même. Nous présentons, dans la suite, les spécificités de la langue arabe pour l'analyse d'opinions.

2.5.1 Catégories d'arabe

Les différentes catégories d'arabe et leurs utilisations conjointes rendent la tâche d'AOA difficile. En effet, l'arabe dispose principalement de trois catégories : l'arabe classique, l'arabe moderne standard, et l'arabe dialectal. Sur le web, ASM et AD sont principalement utilisés. Ceci n'empêche pas que l'AC soit également présent sur le web avec l'utilisation des extraits du Coran et des versets dans le but d'exprimer tout simplement une opinion ou la justifier.
Chaque variante d'arabe dispose donc de ses propres complexités. Ceci signifie implicitement que chaque variante devrait être traitée séparément. Autrement dit, une phase d'identification de variantes pourrait être utile et chaque variante devrait disposer probablement de ses propres outils TALN performants pour garantir l'obtention de bonnes performances.
Des fautes d'orthographe, quasi-présentes dans les différents commentaires sur les réseaux sociaux

quelque soit la variante d'arabe employée, bruitent le sens et voilent la bonne signification. La sémantique peut être ainsi cachée partiellement ou totalement. Les fautes d'orthographe perturbent ainsi la détection de polarité.

2.5.2 Dialecte arabe

Les recherches en ASM sont très avancées par rapport à l'arabe dialectal. Il s'en suit une bonne performance des outils TALN dédiés à l'ASM mais pas celles de l'AD, les recherches étant à leur début et attirant l'attention académique et industriel.
L'arabe dialectal, couramment utilisé sur les plateformes de médias sociaux, fait l'objet d'analyse d'opinions en ligne. L'AD partage pratiquement des caractéristiques de l'ASM, à savoir la richesse morphologique et l'agglutination, mais pas avec le même degré de complexité (absence du duel en AD par exemple).
L'AD dispose d'une grande variété de dialectes qui diffèrent selon la localisation géographique. Chaque dialecte a son propre lexique, ses règles linguistiques appropriées, en plus de ses propres idiomes et ses proverbes particuliers. Malgré le fait que tous les dialectes sont dérivés de l'ASM et partagent donc un peu de vocabulaire, des mots ou expressions communs entre deux dialectes peuvent avoir des polarités totalement différentes. Par exemple, يعطيك العافية est un compliment avec une polarité positive qui signifie "Que Dieu vous accorde la santé" dans le dialecte levantin, alors que cette même phrase a une signification agressive "bruler avec le feu" dans le dialecte tunisien. Considérant ces variantes, un système AOA qui cible un dialecte peut ne pas être efficace pour un autre, car il est développé avec des outils dépendants du dialecte tels que l'analyseur morphologique, les mots vides, les termes de négation et les lexiques polarisés. Est-il mieux de concevoir un système par groupe de dialectes ? comment faire pour la portabilité à d'autres groupes dialectaux ? Serait-il simple de mettre en place un seul système performant pour tous les dialectes ? Toutes ces questions reflètent la difficulté de la tâche.

2.5.3 Polysémie

Il est important de mentionner les différents sens possibles que peut signifier un mot. Chaque sens ne peut être distingué qu'en se basant sur son contexte linguistique, culturel et voire même historique. En effet, autre que le sens propre (sens premier), certains mots peuvent avoir un sens figuré. Le sens propre est relatif aux aspects concret et logique du mot, alors que le sens figuré permet d'interpeller l'imaginaire du lecteur et de provoquer une émotion ou une idée qui ne pourrait pas être exprimée ou dévoilée avec l'unique emploi de sens propre. C'est-à-dire, le sens figuré du mot permet de réaliser des figures de style afin de stimuler et provoquer le lecteur. Le sens figuré peut être exprimé dans n'importe quelle catégorie d'arabe. Ainsi, la langue arabe, quelque soit la catégorie, est caractérisée par les emplois du langage figuratif qui résulte de la liberté accordée au locuteur pour construire de nouvelles images dans le but d'attirer l'attention du lecteur sur le message transmis. Cette richesse de construction d'images exceptionnelles est établie par l'utilisation de sens figurés de

mots. Cependant, l'opposition entre sens propre et sens figuré relève souvent le problème de la polysémie [ZOUAGHI *et al.*, 2004] qui signifie qu'un *même mot recouvre généralement plusieurs sens apparentés néanmoins différents*. Nous citons, à titre d'exemple, le verbe polysémique طار /tar/ qui peut avoir différentes interprétations possibles selon le contexte : déplacement aérien (طار العصفور l'oiseau a volé) ou sentiment fort (طار من شدّة الفرح il a sauté de joie).

2.5.4 Non voyellation

Il faut noter l'impact des voyelles, une caractéristique spécifique à l'arabe, sur la tâche d'analyse d'opinions. En effet, un mot non voyellé peut avoir des significations distinctes avec différentes voyellations possibles [5]. Ces homologues pourraient avoir des polarités différentes. Le tableau 2.12 expose un exemple d'homologue en arabe. En fait, le mot non voyellé جمل pourrait signifier جَمَلٌ (chameau) ou جُمَلٌ (des phrases) avec la polarité neutre, et جَمَّلَ avec la polarité positive. Les voyelles semblent être utiles pour la détection de polarité. Cependant, la majorité de document en ASM ou AD ne sont pas voyellés. La bonne voyellation de mot est guidée par son contexte. L'ajout de voyelles pourrait cerner les sens de mots et simplifier la tâche d'AO. Cependant, les travaux existants dans le cadre de la voyellation automatique de l'arabe ne sont pas nombreux et les systèmes conçus se sont pas performants [Laroussi & Bourouba, 2018].

Mot	Voyellation	Translitération	Traduction	Polarité
جمل	جُمَلٌ	/joumalun/	phrases	neutre
	جمل	/jamalun/	chameau	neutre
	جمّل	/jammala/	embellir	positive

TABLE 2.12 – Différentes voyellisations possibles du mot جمل et leurs polarités correspondantes.

2.5.5 Agglutination

Comme nous l'avons vu dans la section précédente, l'arabe est une langue morphologiquement complexe. Cette complexité nécessite la mise au point de systèmes appropriés capables de gérer la tokenisation, le stemming, la lemmatisation, *etc*. En ce qui concerne l'agglutination, un processus de tokenisation permet de séparer les clitiques de la forme fléchie qui est porteur de sens dans la majorité des cas. Par exemple, les mots الجمال (la beauté), جمالهم (leur beauté), بجمالكم (avec votre beauté) partage la même forme fléchie جمال (une beauté) portant le sens de la beauté avec la polarité positive. Il y en a de même pour la richesse morphologique, les verbes conjugués تعجب, يعجبان, يعجبون partagent le même lemme يعجب.

5. 74% de mots non voyellé accepte plus qu'une voyellation lexicale [Debili & Achour, 1998].

2.5.6 Entitées nommées

Une grande partie de noms propres en arabe est associée aux noms et adjectifs positifs. Par exemple, le prénom سعيد correspond à l'adjectif سعيد qui signifie "heureux", le prénom جميل est en rapport avec l'adjectif جميل (beau), le prénom فرح correspond au nom فرح (une joie) et le prénom جمال fait référence au nom جمال (une beauté). Ceci peut perturber la détection de polarité.
Serait-il nécessaire dans ce cas d'intégrer un module de reconnaissance d'entités nommées ? En arabe, ce dernier n'est pas aussi simple qu'en français. En effet, les noms propres arabes ne commencent pas par majuscule comme dans les langues latines, ce qui complique l'identification des entités nommées en arabe.

2.5.7 Négation

La négation en arabe est exprimée à l'aide de mots de négation spécifiques tels que : "لم", "لن" et "لا". La négation doit être détectée et traitée avec précision, car elle peut convertir le sens d'une phrase donnant une polarité opposée. Par exemple, la phrase (i) احبّ هذا الكتاب (j'aime ce livre) porte une polarité positive et la phrase (ii) لا احبّ هذا الكتاب (je n'aime pas ce livre). Ainsi le fait d'ajouter le terme de négation لا inverse complètement la polarité. Cette tâche devient plus difficile avec l'AD. Les mots de négation sont très différents des mots de l'ASM officiels et ont plusieurs significations telles que le mot "مش" signifiant "non" dans le dialecte tunisien qui peut être utilisé pour une négation (ex : الفطور مش حاضر) ou une question (ex : مش ، تجي غدوة) qui peut induire en erreur le système d'analyse d'opinions.

2.5.8 Système d'écriture

Le système d'écriture de l'arabe pourrait compliquer plusieurs tâches TAL, entre autre la tâche d'analyse d'opinions. En effet, l'utilisation de l'arabizi nécessite une façon de traitement autre que celle dédiée aux caractères arabes. En Arabizi, par exemple, l'analyse de l'énoncé (a) "elfilm moch 7lou" (le film n'est pas beau) doit se baser sur un vocabulaire (ou lexique) en arabizi. L'équivalent de (a) en caractères arabes est (aa) الفلم مش حلو.
Ce problème au niveau d'écriture n'est pas trop compliqué. En fait, une codification *code switching* peut ramener le problème d'analyse en arabizi à un problème d'analyse en arabe, ou inversement. Il suffit d'avoir les ressources nécessaires (lexique ou corpus) pour l'analyse d'opinions.

2.6 Conclusion

Le langage humain est complexe. Apprendre à une machine comment analyser les nuances grammaticales et/ou culturelles, l'argot et les fautes d'orthographe est un processus difficile. Ce chapitre se situe dans le cadre de l'analyse du langage et aborde en particulier la langue arabe et ses spécificités pour l'analyse d'opinions.

Nous nous sommes intéressés dans un premier temps à présenter les caractéristiques de la langue arabe, tout en exposant les spécificités de l'arabe standard moderne et l'arabe dialectal. Ensuite, nous avons présenté les différents prétraitements TALN possibles pour la langue arabe (segmentation, lemmatisation, stemming et light stemming). Puis, nous avons présenté les méthodes de construction de ressources (lexiques et corpus) nécessaires pour l'analyse d'opinions en arabe, et nous avons présenté un survol des travaux existant en AOA. Enfin, nous avons mis l'accent sur certaines spécificités de la langue arabe qui sont en liaison directe avec la tâche d'analyse d'opinions et leur prise en compte pour aider dans la détection d'opinions et l'amélioration des performances.

Dans cette thèse, nous utilisons des réseaux de neurones pour concevoir un système d'AOA. Nous présentons dans le chapitre suivant, les notions de base des réseaux de neurones.

Chapitre 3

Réseaux de neurones pour l'analyse d'opinions en arabe

Sommaire

3.1 Réseaux de neurones : généralités 52
3.1.1 Définition 52
3.1.2 Apprentissage 54
3.1.3 Résumé 57
3.2 Réseaux de neurones convolutifs 58
3.2.1 Entrée du CNN 59
3.2.2 Convolution 60
3.2.3 Pooling 62
3.2.4 Couches entièrement connectées 63
3.2.5 Sortie du CNN 63
3.3 Réseaux de neurones récurrents 64
3.3.1 Présentation des RNN 64
3.3.2 Long-Short Term Memory 68
3.3.3 Long-Short Term Memory Bidirectionnels 69
3.4 Représentations continues de mots 69
3.4.1 Méthodes de construction 70
3.4.2 Techniques d'évaluation 74
3.5 Réseaux de neurones pour l'AO en arabe : état de l'art 75
3.5.1 Méthodes neuronales 75
3.5.2 Prétraitements 78
3.5.3 Synthèse 79
3.6 Conclusion 80

Ce chapitre présente les architectures basées sur les réseaux de neurones artificiels connaissant un grand essor aujourd'hui et notamment, durant les dernières années, un succès remarquable dans la tâche d'analyse d'opinions.
Quoiqu'ils remontent à la fin des années 50 [Rosenblatt, 1957, 1958], les réseaux de neurones artificiels ont permis, cette dernière décennie, de franchir des paliers dans plusieurs applications. Les architectures neuronales ont révélé leur potentiel grâce à la disponibilité des dispositifs de calcul puissants (tels que les processeurs graphiques) pour paralléliser massivement les calculs au moment de l'apprentissage.

Ce chapitre présente, de façon succincte, les architectures neuronales les plus utilisées pour l'AOA. Nous exposons, dans un premier temps, les architectures neuronales utilisées dans cette thèse. Nous présentons, dans un deuxième temps, les représentations continues de mots utilisées en entrées des réseaux de neurones. Ceci nous conduit à présenter un survol de l'état de l'art des méthodes utilisées pour la construction de ces représentations ainsi que leurs techniques d'évaluation.

3.1 Réseaux de neurones : généralités

Avant d'aborder la modélisation de textes avec ces architectures neuronales, nous présentons, dans cette section, les concepts de base des réseaux de neurones artificiels : neurone formel, apprentissage et rétro-propagation.

3.1.1 Définition

Les premiers réseaux [Rosenblatt, 1958] ont éclos sous le nom de perceptron. Ce dernier, composé d'un seul neurone, représente la forme la plus simple d'un réseau de neurones, et ne permet de résoudre que des problèmes linéairement séparables. De nouveaux types de réseaux artificiels [Widrow & Hoff, 1960; Hopfield, 1982; Rumelhart *et al.*, 1988; LeCun *et al.*, 1990, 2010] ont été conçus pour résoudre des problèmes de classification complexes [Minsky, 1969].

3.1.1.1 Neurone formel

Un réseau de neurones artificiel se compose d'un ensemble d'éléments appelés "neurones formels" organisés sous forme de graphe plus au moins complexe. Un neurone formel est une analogie d'un neurone biologique, comme le montre la figure 3.1. Cette figure illustre la ressemblance entre un neurone formel et un neurone biologique.

Un neurone formel prend des entrées $X = (x_1, x_2, \ldots, x_n)$ auxquelles sont associées des poids $W = (\omega_1, \omega_2, \ldots, \omega_n)$ qui reflètent l'importance de l'information véhiculée x_i où $1 \leqslant i \leqslant n$. Ce neurone prend en entrée également un biais (b). Ce dernier permet d'ajouter de la flexibilité au réseau en agissant sur la position de la frontière de décision [Rosenblatt, 1957].
Le neurone formel fournit en sortie y qui peut être utilisée comme entrée pour d'autres neurones. La sortie y correspond à l'application d'une fonction d'activation φ sur la somme des vecteurs d'entrées

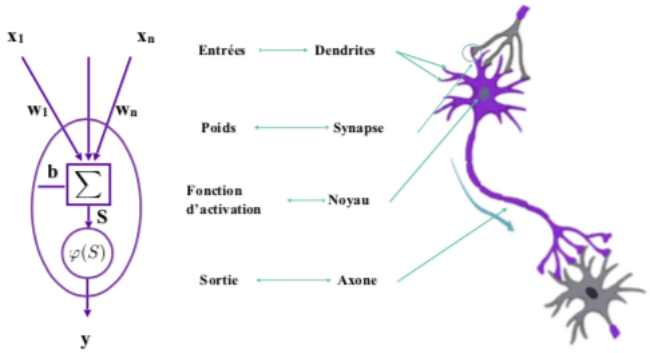


FIGURE 3.1 – Analogie entre un neurone formel et un neurone biologique [Ghannay, 2017].

$X = (x_1, x_2, \ldots, x_n)$ pondérés par le vecteur de poids $W = (\omega_1, \omega_2, \ldots, \omega_n)$ et le biais (voir équation 3.2).

$$y = \varphi(W.X + b) \tag{3.1}$$

$$= \varphi(\sum_{i=1}^{n} \omega_i x_i + b) \tag{3.2}$$

La fonction d'activation φ opère une transformation d'une combinaison affine des entrées $Z = W.X + b$ et calcule l'état du neurone (activé ou non). Elle est souvent définie par une des fonctions suivantes :

— Identité (linéaire) :

$$\varphi(Z) = Z \tag{3.3}$$

— Sigmoïde :

$$\varphi(Z) = \frac{1}{1 + e^{-Z}} \tag{3.4}$$

— Tangente hyperbolique :

$$\varphi(Z) = \frac{1 - e^{-Z}}{1 + e^{-Z}} \tag{3.5}$$

— ReLu (Rectified Linear unit) :

$$\varphi(Z) = max(0, Z) \tag{3.6}$$

— Softmax :

$$\varphi(Z) = \frac{e^{-Z_j}}{\sum_{k=1}^{K} e^{Z_k}} \qquad \text{pour tout j } \in \{1, \ldots, K\} \tag{3.7}$$

3.1.1.2 Perceptron multi-couches

Un perceptron multi-couches (*Multilayer Perceptron* MLP) [Minsky, 1969] est un réseau de neurones à propagation avant (*feed-forward*), dans lequel l'information est véhiculée dans un seul sens : de l'entrée vers la sortie (voir figure 3.2). Le MLP est constitué de cascade de couches : une couche

d'entrée, une ou plusieurs couches cachées et une couche de sortie. Les couches voisines sont complètement connectées, c'est-à-dire, les neurones de chaque couche sont liés à tous les neurones de la couche précédente et à tous ceux de la couche suivante. Par contre, aucune connexion n'existe entre les neurones d'une même couche.

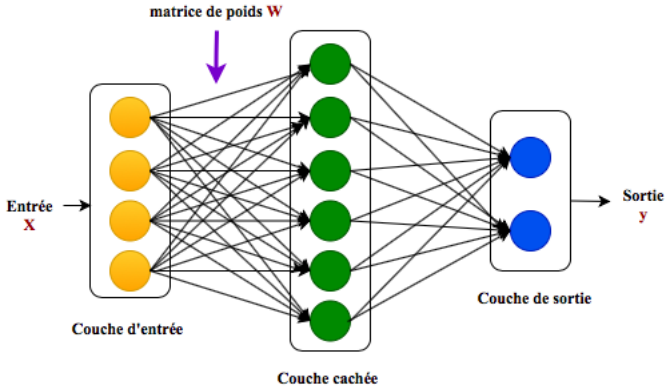


FIGURE 3.2 – Schéma d'un perceptron multi-couches (avec une seule couche cachée).

Le réseau MLP propage l'information de la couche d'entrée vers la (les) couche(s) cachée(s) et enfin la couche de sortie donnant un score pour chaque neurone de sortie. Le score le plus élevé sera retenu et la classe correspondante sera prédite. Cette dernière représente l'*hypothèse* du réseau.

3.1.2 Apprentissage

L'apprentissage supervisé du réseau de neurones consiste à ajuster ses paramètres (matrices de poids et bais) pour classer un ensemble d'exemples (des documents d'opinions dans le cas de l'AO) en se basant sur un corpus d'apprentissage regroupant n exemples et leurs classes correspondantes (des documents et leurs polarités associées). Les algorithmes d'apprentissage dépendent de la tâche, des données d'apprentissage, des architectures neuronales, *etc*. Pour les problèmes de classification binaire, la fonction sigmoïde (équation 3.4) peut être utilisée. Par contre, pour les problèmes de classification multi-classes, la fonction Softmax (équation 3.7) est souvent la plus utilisée pour représenter une distribution de probabilités sur K classes différentes.

La rétro-propagation du gradient (*backpropagation*) [Rumelhart *et al.*, 1988] représente l'algorithme d'apprentissage le plus utilisé. Cet algorithme adapte les poids afin de minimiser la différence entre la sortie du réseau (l'hypothèse) y et la sortie désirée (la référence) s. Cette différence est représentée par la fonction de coût (*Loss* L), dite aussi fonction d'erreur. Ce qui revient à minimiser la fonction de coût par calcul du gradient de l'erreur pour chaque neurone du réseau, de la couche de sortie à la première couche cachée. Normalement, la fonction de coût est non linéaire [Stemmer *et al.*, 2002], elle peut être, par exemple, l'erreur quadratique moyenne (*Mean Square Error* MSE)

(équation 3.8) ou l'entropie croisée (*Cross Entropy* CE) (équation 3.9).

$$L = \frac{1}{n}\sum_{i=1}^{n}(s_i - y_i)^2 \tag{3.8}$$

$$L = \frac{1}{n}\sum_{i=1}^{n}(s_i log(y_i) - (1-s_i) log(1-y_i)) \tag{3.9}$$

L'algorithme de la rétro-propagation du gradient commence par la phase de "propagation avant" (*forward pass*) au cours de laquelle les valeurs de sortie du réseau $y_i, 1 \leqslant i \leqslant n$ sont calculées ainsi que la fonction de coût L entre les hypothèses et les références. Ensuite, l'algorithme effectue la phase de "propagation arrière" (backward pass) . Cette dernière consiste à rétro-propager la dérivée partielle du coût ($\frac{\partial L}{\partial W}$) par rapport aux poids W du réseau. Enfin, les poids sont mis à jour en fonction de cette dérivé partielle selon l'équation 3.10 :

$$W' = W - \Delta W \tag{3.10}$$

avec :

$$\Delta W = -\eta \frac{\partial L}{\partial W} \tag{3.11}$$

où $\eta > 0$ représente le taux d'apprentissage (*learning rate*), W est la matrice de poids initiale (utilisée lors de la propagation avant), et $W^{'}$ est la matrice de poids mise à jour.

Le taux d'apprentissage constitue un hyper-paramètre du réseau de neurones et détermine la vitesse de convergence d'apprentissage. Un taux η trop élevé augmente la chance d'engendrer des oscillations entre des minimums, mais un taux trop faible rend la convergence lente et risque de rester coincé dans un minimum local. Cette quantité peut être fixe ou variable. Certaines méthodes existent pour faire varier η de façon optimale durant l'apprentissage (en faisant modifier la valeur de η au fil des itérations/époques) avec une adaptation du taux d'apprentissage (*adaptative learning rate*). Trois types d'adaptation sont proposés :

— Variation du taux d'apprentissage au cours de l'apprentissage : cela permet d'accélérer et améliorer la convergence. On peut citer par exemple AdaDelta [Bottou, 2010], AdaGrad [Duchi *et al.*, 2011], Adam [Kingma & Ba, 2014], *etc*.

— Prise en considération de la mise à jour précédente dans le calcul de la mise à jour de poids actuelle (voir équation 3.12) par introduction de la notion de *momentum* $\alpha \in [0,1]$ [Plaut *et al.*, 1986]. Il existe plusieurs variantes de l'algorithme de momentum [Sutskever *et al.*, 2013; Zhang *et al.*, 2015].

— Combinaison des deux manières d'adaptation précédentes. Cela consiste à considérer la notion de momentum pendant l'apprentissage. On cite par exemple [Dozat, 2016].

$$\Delta W(t) = -\eta \frac{\partial L}{\partial W(t)} + \alpha W(t-1) \tag{3.12}$$

Avec les réseaux de neurones, trois stratégies d'apprentissage par rétro-propagation sont souvent utilisées :

— La rétro-propagation stochastique (*stochastic backpropagation*) consiste à mettre à jour les poids à chaque présentation d'un exemple d'apprentissage.
— La rétro-propagation par lot (*batch backpropagation*) consiste à mettre à jour les poids selon la moyenne de la fonction de coût sur tous les exemples d'apprentissage qui constituent un lot.
— La rétro-propagation par mini-lot (*minibatch backpropagation*) consiste à mettre à jour les poids après présentation de chaque mini-lot (un groupe d'exemples d'apprentissage dont le cardinal est inférieur à celui du lot). Elle est la plus utilisée car elle combine les deux stratégies précédentes.

Il est important de mélanger les exemples d'apprentissage. En effet, la convergence risque d'être lente avec beaucoup d'exemples consécutifs de la même classe. Pour cette raison, il est préférable de mélanger aléatoirement les exemples afin d'éviter la corrélation entre les exemples consécutifs.

La phase d'apprentissage est coûteuse. En effet, l'apprentissage des réseaux de neurones consiste à appliquer l'algorithme d'apprentissage plusieurs fois appelées *époques*, elles-même constituées d'*itérations*. Une itération est une mise à jour des poids. Une époque est l'application de l'algorithme sur tous les exemples d'apprentissage. À chaque époque, le but est de minimiser la fonction de coût L. Le nombre d'époques représente un hyper-paramètre du réseau. Le choix du nombre d'époques est important afin de garantir la convergence, c'est-à-dire l'obtention d'une valeur de L nulle, ce qui n'est pas toujours possible. Il est important que le nombre d'époques soit assez grand pour converger mais cela peut conduire à un temps d'apprentissage très lent et éventuellement un sur-apprentissage (*over-fitting*). Pour pallier ce problème, une *patience* en nombre d'époques peut être fixée. Cette patience conditionne l'arrêt d'apprentissage : si le système ne s'améliore pas sur un corpus de validation (distinct du corpus d'apprentissage et de test) en un nombre d'époques égal à la patience, alors il arrête l'apprentissage. Le compteur de la patience est réinitialisé après à chaque amélioration.

La convergence du réseau vers le minimum est important, mais ceci ne garantit pas une bonne généralisation et ne prévient pas un sur-apprentissage. Plusieurs méthodes pour éviter le sur-apprentissage du réseau existent. Nous présentons, dans la suite, la régularisation et le *dropout*.
La régularisation $L1$ et $L2$ [Collobert & Bengio, 2004; Bengio *et al.*, 2006] consiste à ajouter un terme, dit de régularisation, à la fonction du coût L. Ce terme contrôle les poids pendant l'apprentissage en pénalisant ceux ayant de grandes valeurs. Les termes de régularisation les plus connus sont : $L1$ appelé aussi *lasso regression* (somme des valeurs absolues de poids) et $L2$, dit aussi *ridge regression* (somme des carrés de poids). Ces termes ($L1$ et $L2$) sont pondérés par des coefficients [Bengio, 2012].
Le décrochage (*dropout*) [Srivastava *et al.*, 2014] est une autre technique pour éviter le sur-apprentissage dans les réseaux de neurones. Il consiste à désactiver temporairement une partie des neurones et leurs connections pendant l'apprentissage, comme illustré dans la figure 3.3. La valeur du dropout représente la partie ou le nombre de neurones à désactiver. Le dropout limite la co-adaptation sur les données d'apprentissage et donne au réseau la possibilité de trouver de nouveaux moyens pour résoudre le même problème [Deshpande, 2016]. En pratique, le dropout se fait en multipliant la sortie

des neurones retirées par 0.

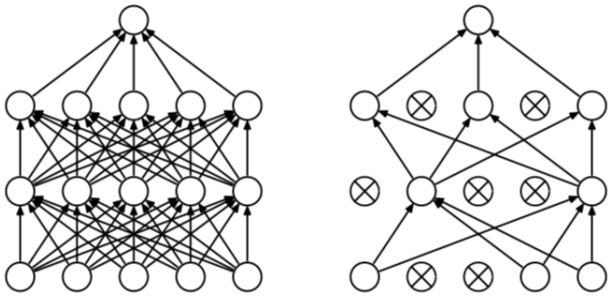


FIGURE 3.3 – Application du dropout dans un réseau de neurones [Srivastava *et al.*, 2014].

3.1.3 Résumé

Les réseaux de neurones artificiels disposent de paramètres et d'hyper-paramètres. Les matrices de poids et les biais constituent les paramètres. Les hyper-paramètres sont des variables spécifiques dont les valeurs sont définies avant le début d'apprentissage, c'est-à-dire ils ne sont pas appris lors de l'apprentissage du réseau mais doit être fixés par l'utilisateur. Parmi ces hyper-paramètres, nous citons principalement :

- Nombre de couches cachées et le nombre de neurones par couche cachée
- Fonction d'activation
- Paramètres de régularisation
- Nombre d'époques d'apprentissage
- Early stopping qui permet d'arrêter l'apprentissage lorsque le score d'évaluation sur le corpus de validation ne s'améliore pas au bout d'un certain nombre d'époques.
- Taille du mini-lot et sa normatisation
- Coefficient de dropout
- Taux d'apprentissage

Il n'y a pas de règles universelles à l'affectation de ces hyper-paramètres. Des études empiriques peuvent être effectuées pour les ajuster. La taille de l'espace de recherche d'hyper-paramètres augmente exponentiellement avec leur nombre. Ceci rend le choix des hyper-paramètres critique.

Les paramètres retenus du réseau de neurones (comme étant les plus performants) sont ceux donnant le meilleur score d'évaluation sur un ensemble de données non utilisé pendant l'apprentissage, appelé corpus de validation (dit aussi corpus de développement). Il est important de noter que la métrique d'évaluation utilisée sur le corpus de validation est différente de la fonction de coût utilisée pour l'apprentissage, et dépend de la tâche. Le paramètrage du système finalement conservé est celui ayant donner la meilleure performance sur le corpus de validation.

3.2 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (*Convolutional Neural Network* CNN) sont un type spécialisé de réseaux de neurones multi-couches généralement utilisés quand l'entrée est structurée selon une grille (ex : une image). Ces réseaux ont été inspirés des travaux de [Hubel & Wiesel, 1962] sur le cortex visuel des animaux, et plus particulièrement sur ses propriétés : les champs récepteurs locaux et le partage de poids. Les CNN sont initialement introduits par [Fukushima, 1980] pour une tâche de reconnaissance de formes, et ont été popularisés, dans les années 1990, avec les travaux de [LeCun *et al.*, 1990] sur la reconnaissance de caractères. La figure 3.4 montre les différentes couches d'un réseau de neurones convolutif. Ce dernier est composé d'un ou plusieurs bloc de convolution et de pooling, une ou plusieurs couches cachées et une couche de sortie. Le CNN prend en entrée une grille multi-dimensionnelle représentant une instance d'apprentissage ou d'inférence, et fournit en sortie la classe correspondante.

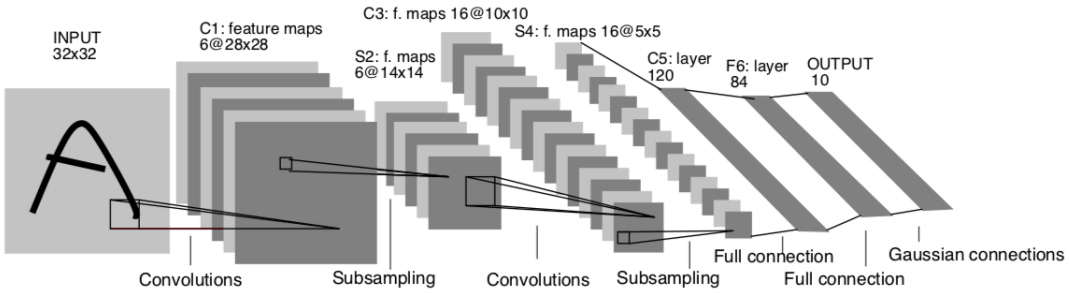


FIGURE 3.4 – Réseau de neurones convolutif composé de deux couches de convolution et de pooling (*subsampling*), suivi de deux couches cachées et d'une couche de sortie [LeCun *et al.*, 1990].

Les approches d'apprentissage automatique standard nécessitent une définition au préalable d'un ensemble de descripteurs adéquats à la tâche de classification. Les performances de ces approches sont fortement conditionnées par la qualité de descripteurs pré-définis utilisés et leurs pertinences à la tâche. Ceci reflète l'importance de la phase de préparation et de sélection de descripteurs : la phase de *features engineering* [Abbasi *et al.*, 2008]. Contrairement aux approches d'apprentissage automatique standard basées sur des descripteurs pré-définis, un CNN est capable de définir et extraire, au moment de l'apprentissage, des descripteurs spécifiques adaptés à la tâche en question. Ceci peut justifier sa performance et sa popularité.

Aujourd'hui, les CNN sont utilisés dans plusieurs applications de traitement d'images [Perez-Munuzuri *et al.*, 1993; Xu *et al.*, 2014], traitement de la parole [Palaz *et al.*, 2015; Dai *et al.*, 2017] et traitement de textes [Collobert *et al.*, 2011b; Kim, 2014a; Lin *et al.*, 2016]. Il existe une différence d'implémentation de CNN en fonction du domaine d'application. En effet, la différence de développement de réseaux de neurones convolutifs pour le traitement d'images et le traitement de langues réside principalement au moment de la convolution. Cette dernière permet d'extraire des descripteurs via des

filtres. En traitement d'images, l'extraction de descripteurs s'effectue sur des petites zones de l'image (entrée du réseau), où chaque filtre se déplace dans deux sens (horizontal et vertical) sur l'image. En traitement de texte, le fonctionnement est différent. En effet, le filtre couvre une séquence de mots et ne se déplace que dans un seul sens. Dans cette thèse, nous utilisons les CNN pour la tâche d'analyse d'opinions au niveau document. C'est pour cela que nous nous intéressons au fonctionnement des CNN pour le traitement de texte. Nous présentons, dans la suite, la structure générale d'un CNN appliqué pour la classification de documents textuels.

3.2.1 Entrée du CNN

En traitement de texte, l'entrée du système représente une séquence de mots sous forme d'une structure unidirectionnelle. Dans le cas d'un réseau de neurones convolutif, [Collobert & Weston, 2008] proposent d'exploiter les représentations vectorielles de mots pour transformer la séquence de mots en une grille de dimension $n \times k$, où n est le nombre de mots et k la dimension des représentations vectorielles de mots, et obtenir ainsi une entrée sous forme de grille bi-dimensionnelle. En pratique, ceci revient à utiliser une couche d'embeddings (*lookup table* Embed). Cette dernière est une matrice d'embeddings $E \in \mathbb{R}^{V \times k}$, où V est la taille du vocabulaire. Chaque ligne de la matrice d'embeddings représente la représentation vectorielle E_ω correspondante à un mot ω du vocabulaire (voir équation 3.13).

$$Embed(\omega) = E_\omega \tag{3.13}$$

Soit une séquence de n mots $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ (une entrée textuelle du CNN), la sortie de la couche Embed est une représentation matricielle $M \in \mathbb{R}^{n \times k}$ définie selon l'équation 3.15.

$$M = Embed(\Omega) \tag{3.14}$$

$$= \begin{pmatrix} E_{\omega 1} \\ E_{\omega 2} \\ \cdots \\ E_{\omega n} \end{pmatrix} \tag{3.15}$$

Les embeddings de la couche Embed sont généralement initialisés soit de façon aléatoire soit à l'aide d'un ensemble d'embeddings pré-entrainés (voir section 3.4). Ces embeddings constituent les paramètres de Embed. Ils sont mis à jour de deux manières différentes au moment de l'apprentissage du CNN [Kim, 2014b]. La première manière consiste à ne pas modifier les embeddings $E_{\omega i}$. Le CNN correspondant est dit *statique*. La deuxième manière permet d'obtenir un CNN *non statique* : les embeddings sont mis à jour à l'aide de l'algorithme de rétro-propagation.

La matrice d'embeddings $M \in \mathbb{R}^{n \times k}$ est de dimension fixe. La dimension d'embeddings k étant fixe, il faut alors fixer n le nombre de mots dans le document d'entrée : la longueur du document. Si la longueur du document est supérieure à n, alors il est nécessaire de tronquer les mots supplémentaires : c'est le tronquement (*truncating*). Si la longueur l est inférieure à n, il est donc nécessaire d'ajouter

$n-l$ fois un symbole spécial hors vocabulaire (par exemple : <PAD> comme dans la figure 3.5). Le vecteur correspondant à <PAD> dans la matrice M est initialisé à zéro. Or, il existe trois façons de faire : soit tronquer/ajouter au début du document (pre), ou à la fin (post), ou sur les extrémités.

La figure 3.5 montre la construction de la matrice M. Elle décrit le processus de transformation d'un document en une représentation matricielle de taille 7×6 à l'aide de Embed. Par exemple, le document $doc = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_1)$ est de longueur $6 < 7$, nous ajoutons alors une fois le symbole <PAD> à la fin du document (post padding). Ensuite une opération de correspondance est effectuée avec Embed pour déterminer la représentation vectorielle de chaque mot ω_i.
Afin de construire la matrice M, la représentation vectorielle de chaque mot ω_i du document est initialisée par son embedding pré-entrainé s'il existe, sinon elle est inisialisée aléatoirement. Dans notre exemple *doc*, les mots $\omega_1, \omega_2, \omega_3, \omega_5$ ont chacun un embedding pré-entrainé dans *Embed*. Ils sont alors initialisés par leurs embeddings pré-entrainés dans la matrice M. Si un mot n'a pas d'embedding pré-entrainé dans *Embed*, il est alors initialisé aléatoirement dans M (c'est le cas du mot ω_4 dans *doc*).

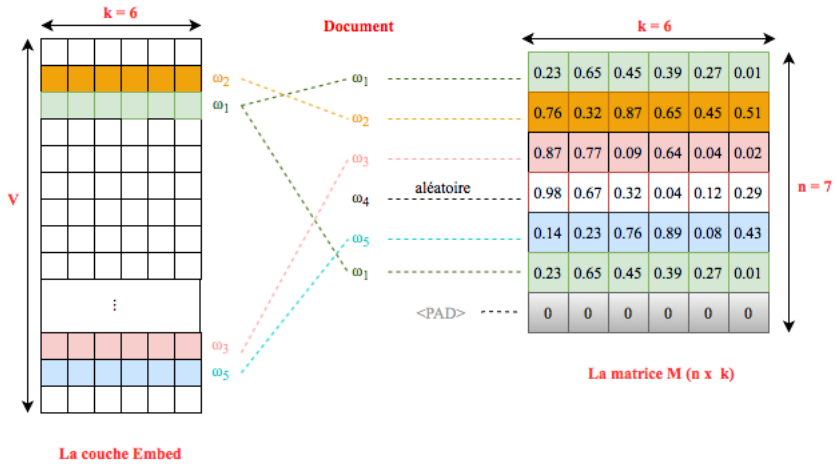


FIGURE 3.5 – Mécanisme de construction de la matrice $M \in \mathbb{R}^{n \times k}$ correspondant à un document en se basant sur la table d'embeddings Embed.

3.2.2 Convolution

La convolution est une opération mathématique fondamentale pour les réseaux de neurones convolutifs. Elle consiste à multiplier, ou *convoluer* la représentation matricielle M par une autre appelée matrice de convolution (ou filtre) pour produire une carte de caractéristiques (*feature map*).

Les paramètres de la couche de convolution consistent en un ensemble de filtres apprenables. Les filtres correspondent à des poids initialisés aléatoirement puis mis à jour avec l'algorithme de rétropropagation du gradient lors de l'apprentissage de CNN. La taille des filtres et leurs sens de direction dépendent de la tâche et de la dimension de la carte de caractéristiques. Pour le traitement de texte, l'opération de convolution est de dimension 1. Le filtre F est de dimension $h_c \times k$, où h_c est le nombre

mots consécutifs à considérer dans le filtre, et k la dimension d'embedding. Le filtre se déplace donc dans un seul sens, du haut en bas comme illustré dans la figure 3.6, avec un pas s_c afin de fournir une carte de caractéristiques de dimension 1.

La couche de convolution implique un ensemble de filtres, des fenêtres de convolution et un pas de déplacement :

— Un ensemble de **filtres** de dimension $h_c \times k_c$, où h_c est un hyper-paramètre fixée par l'utilisateur et k_c la largeur correspondante à la largeur de la matrice d'entrée M ($k_c = k$ [1]).

— Une **fenêtre de convolution** de même dimension que le filtre $h_c \times k$ qui représente une partie de l'entrée $M_{i:i+h_c-1}$ qui s'étend à travers toute l'entrée M.

— Un **pas de déplacement** $s_c \geq 1$ précisant le déplacement de la fenêtre de convolution de haut en bas sur la matrice M.

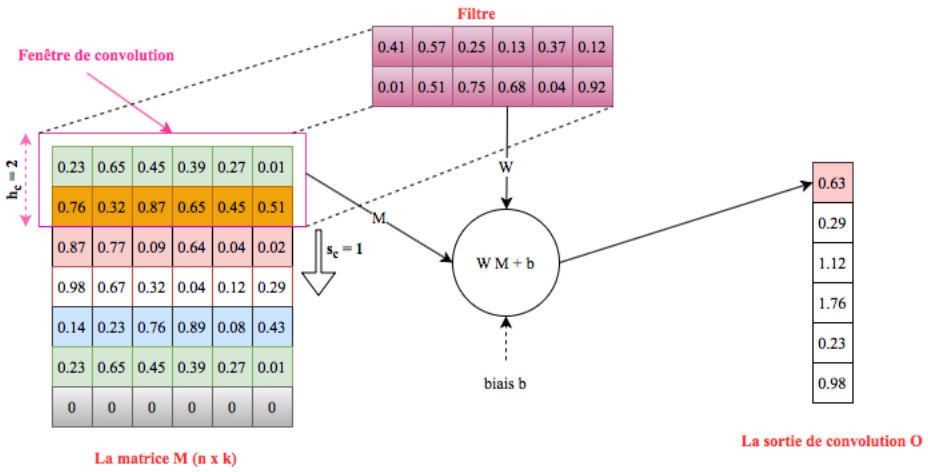


FIGURE 3.6 – Exemple d'application d'une convolution 1D sur une matrice d'embeddings M.

Nous décrivons, dans la figure 3.6, l'opération de convolution appliquée à une matrice $M \in \mathbb{R}^{n \times k}$. Chaque filtre F de dimension $h_c \times k$ est appliqué sur toutes les fenêtres de convolution correspondantes possibles $M_{i:i+h_c-1}$ pour calculer leurs produits de convolution et produire un vecteur de sortie $O = (o_1, o_2, \ldots, o_{Do}) \in \mathbb{R}^{D_o \times 1}$, où D_o est calculé selon l'équation 3.16. Chaque composant o_i ,$1 \leqslant i \leqslant D_o$, du vecteur O est calculé selon l'équation 3.17 (dans notre exemple, D_o est un vecteur de taille $\frac{7-2}{1} + 1 = 6$).

$$D_o = \frac{n - h_c}{s_c} + 1 \tag{3.16}$$

$$o_i = F \times M_{i:i+h_c-1} + b \quad , \quad 1 \leqslant i \leqslant D_o \tag{3.17}$$

1. Dans le cas de traitement d'images, k_c peut être inférieur à k. La sortie de la convolution O est donc un vecteur à 2 dimensions.

Chaque couche de convolution doit par la suite appliquer une fonction non linéaire φ au vecteur $O = (o_1, o_2, \ldots, o_{Do})$ afin d'extraire les descripteurs appris [Kim, 2014a] et former la carte de caractéristiques $C = (c_1, c_2, \ldots, c_{Do}) \in \mathbb{R}^{D_o \times 1}$, où c_i est calculée avec l'équation 3.19.

$$c_i = \varphi(o_i) \quad , \quad 1 \leqslant i \leqslant D_o \tag{3.18}$$

$$= \varphi(F \times M_{i:i+h_c-1} + b) \tag{3.19}$$

Plusieurs filtres peuvent être appliqués à la même couche de convolution pour produire plusieurs cartes de caractéristiques. Il y aura autant de cartes de caractéristiques que de filtres. Ces cartes sont ensuite transférer vers une couche de *pooling*.

3.2.3 Pooling

Le sous-échantillonnage (*pooling*) est une opération importante pour les réseaux de neurones convolutifs. Il permet d'une part de réduire le nombre de paramètres en réduisant les dimensions des cartes de caractéristiques tout en gardant les informations les plus pertinentes et d'autre part, de respecter l'ordre de mots du document en entrée du CNN. Comme la couche de convolution, un pooling uni-dimensionnnel $1D$ est appliqué pour le traitement de texte.

L'opération de pooling $1D$ est constituée d'une fenêtre de pooling de hauteur h_p se déplaçant sur chaque carte de caractéristiques $C \in \mathbb{R}^{D_o \times 1}$ dans un seul sens du haut en bas avec un pas de déplacement s_p, comme illustré dans la figure 3.7. La sortie de la couche de pooling est un vecteur $\hat{C}$ de taille D_p, où chaque élément de $\hat{C}$ est obtenu par application de l'opération de pooling sur une fenêtre $C_{i:i+h_p-1}$ de chaque carte C. La valeur de D_p est calculée selon la formule 3.20.

$$D_p = \frac{D_o - h_p}{s_p} + 1 \tag{3.20}$$

Généralement, trois types d'opérations de pooling peuvent être utilisées : le *max-pooling*, le *min-pooling* et l'*avg-pooling*.

- **Max-pooling** : consiste à retourner la valeur maximale des valeurs locales de chaque fenêtre de pooling.
- **Min-pooling** : consiste à retourner la valeur minimale des valeurs locales de chaque fenêtre de pooling.
- **Avg-pooling** : consiste à retourner la valeur moyenne des valeurs locales de chaque fenêtre de pooling.

Le *max-pooling* est le plus utilisé pour des problèmes de classification TALN vu qu'il tend à donner plus d'importance aux mots les plus discriminants [Collobert & Weston, 2008]. Alors que l'*avg-pooling* a pour conséquence un lissage peu approprié sur l'ensemble des mots d'un document.

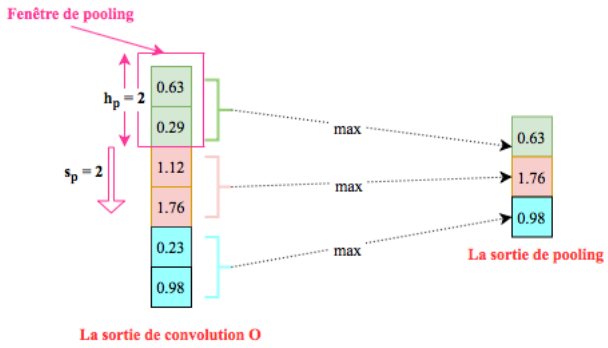


FIGURE 3.7 – Fonctionnement d'une opération de *Max-pooling* appliquée sur une carte de caractéristiques.

Nous illustrons dans la figure 3.7 le principe de fonctionnement de l'opération de pooling. Étant une fenêtre de pooling de hauteur $h_p = 2$ et un pas de déplacement $s_p = 2$, l'opération de max-pooling est appliquée, du haut vers le bas, sur une carte de caractéristiques $C \in \mathbb{R}^{6\times1}$. Il consiste à récupérer la valeur maximale locale de chaque fenêtre de pooling, produisant ainsi en sortie D_p de taille $\frac{6-2}{2}+1 = 3$ unités. Un vecteur $\hat{C} \in \mathbb{R}^{3\times1}$ représente donc la sortie de la couche de pooling.

3.2.4 Couches entièrement connectées

Dans un réseau de neurones convolutif, les couches entièrement connectées (*Fully Connected* FC) constituent une modélisation de haut niveau : c'est une abstraction des entrées. Elles permettent de déterminer le lien entre les représentations abstraites et la sortie du réseau. La première couche FC après la couche de pooling correspond à une concaténation des sorties $\hat{C}$. Elle peut être éventuellement suivie par d'autres couches cachées. Les neurones de chaque couche FC sont liés à tous les neurones de la couche précédente et à tous ceux de la couche suivante. Par contre, aucune connexion n'existe entre les neurones d'une même couche. Chaque neurone est activée par une fonction φ détaillée dans la section 3.1.1.1.

3.2.5 Sortie du CNN

La dernière couche du réseau est une couche FC et représente sa couche de sortie. Elle permet de prédire une classe ou une valeur continue selon la nature de la tâche : une régression ou une classification.

— Régression : la couche de sortie est formée d'un seul neurone produisant une valeur réelle dans $]-\infty,+\infty[$.

— Classification : le nombre de neurones de la couche de sortie est égal au nombre de classes. Deux fonctions d'activation sont généralement utilisées. Dans le cas d'une classification binaire, la fonction *sigmoïde* (voir équation 3.4) est utilisée afin d'attribuer une valeur réelle dans l'intervalle $[0,1]$. Si la valeur est inférieure à 0.5, alors la classe est $\hat{y} = 0$ est prédite. Sinon, la classe prédite est $\hat{y} = 1$. Dans une classification multi-classes, la fonction *softmax* (voir

équation 3.7) représente une distribution de probabilités sur K classes différentes. La classe prédite $\hat{y}$ est calculée selon l'équation 3.21 en appliquant la fonction *argmax* sur la sortie de *softmax*.

$$\hat{y} = \underset{i \in \{1,\ldots,K\}}{\arg\max} \ (\text{softmax}(Z)) \tag{3.21}$$

3.3 Réseaux de neurones récurrents

Nous introduisons, dans cette section, les réseaux de neurones récurrents : origine, types et mode de fonctionnement. Nous présentons, par la suite, les deux variantes de type LSTM et BiLSTM .

3.3.1 Présentation des RNN

Un réseau de neurones récurrent (*Recurrent Neural Network* RNN), présenté dans la figure 3.8, comporte des cycles au sein du réseau de neurones [Elman, 1990]. Il utilise l'information d'autres éléments de la séquence d'entrée supposant la liaison entre les différents éléments de la séquence [Medsker & Jain, 1999; Mikolov *et al.*, 2011], et non pas seulement des éléments isolés n'ayant pas de liaison avec les autres éléments constituant la séquence. La motivation principale derrière ce type d'architectures neuronales et de manipuler des séquences d'observation en tenant compte de l'aspect temporel qui est traduit par l'ordre d'apparition des mots dans la séquence textuelle.

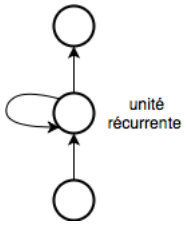


FIGURE 3.8 – Représentation compacte d'un RNN. La flèche pointillée reflète la récurrence sur tous les éléments de la séquence d'entrée.

L'aspect récurrent dans les réseaux RNN consiste à considérer lors de l'étape actuelle, une information extraite lors d'une autre étape (précédente ou suivante). Ce qui permet de conserver des informations sur le contexte en considérant des éléments provenant d'autres étapes. Cette liaison permet au RNN d'encoder des dépendances latentes entre les éléments d'une séquence d'entrée tout en respectant leurs ordres d'apparition dans la séquence. Ceci est assuré par des noeuds cachés récurrents, appelés *noeuds de contexte*, et permet ainsi de traiter une séquence d'entrée de longueur variable. Chaque noeud s'exécute en faisant appel à l'état du réseau du noeud précedent (un appel réseau).

L'idée des réseaux récursifs décrite précédemment n'implique pas une longueur de phrase fixe et donc le nombre d'appels réseau. Dans ce cas, nous devons utiliser une méthode d'apprentissage spéciale appelée propagation à travers le temps (BPTT) [Williams & Zipser, 1995; Gers *et al.*, 2002].

Cet algorithme est dérivé de l'algorithme de rétro-propagation classique pour les réseaux de neurones sauf que le gradient est propagé vers l'arrière nécessitant une mémoire importante.

Trois variantes de RNN peuvent être répertoriées : RNN de type Elman, Jordan et combiné. La figure 3.9 illustre ces trois variantes.

— RNN de type Elman : la récurrence consiste à considérer dans le noeud courant l'information extraite d'un autre noeud caché [Elman, 1990].

— RNN de type Jordan : la récurrence consiste à considérer dans le noeud courant l'information extraite de la couche de sortie principale du réseau [Jordan, 1997].

— RNN de type Elman et Jodan combiné : la récurrence est à la fois de type Elman et Jordan. Elle consiste à considérer les informations extraites d'un noeud caché et de la couche de sortie du RNN.

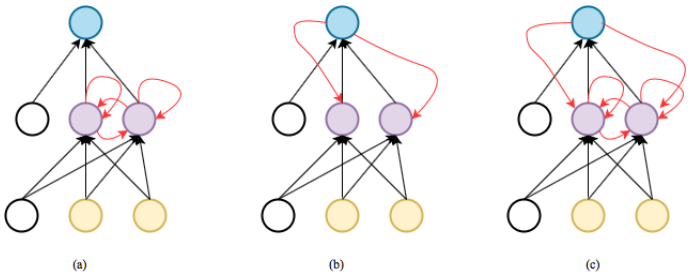


FIGURE 3.9 – Schémas des variantes de RNN : (a) RNN de type Elman, (b) RNN de type Jordan et (c) RNN de type Elman et Jordan combiné.

Nous nous référons dans la suite à la variante Elman. Deux types de récurrence peuvent être distingués : récurrence avant et récurrence arrière. Soit un RNN qui prend en entrée une séquence de mots $x = (x_1, x_2, \ldots, x_n)$ et définit la séquence de noeuds cachés $h = (h_1, h_2, \ldots, h_n)$ pour fournir la séquence de sortie $y = (y_1, y_2, \ldots, y_n)$ en étirant de $t = 1, \ldots, n$ selon les formules 3.22 et 3.23 en cas de RNN avant et les formules 3.24 et 3.25 en cas de RNN arrière, où φ est une fonction d'activation (pas nécessairement la même fonction pour le calcul de h_t et y_t), b_t est le biais du noeud courant, et b_y est le biais du noeud de sortie. Nous présentons, dans la suite, les RNN avant et arrière.

a) RNN avant :

Un RNN avant *forward* consiste à récupérer l'information du noeud caché précédent pour calculer la sortie du noeud caché courant : il parcourt la séquence d'entrée de gauche à droite. De cette façon, le réseau bénéficie des informations du passé permettant ainsi de dépasser les capacités d'un simple réseau de neurones à propagation avant. La figure 3.10 illustre un RNN avant.

$$h_t = \varphi(W_h h_{t-1} + W_x x_t + b_t) \tag{3.22}$$

$$y_t = \varphi(W_h h_t + b_y) \tag{3.23}$$

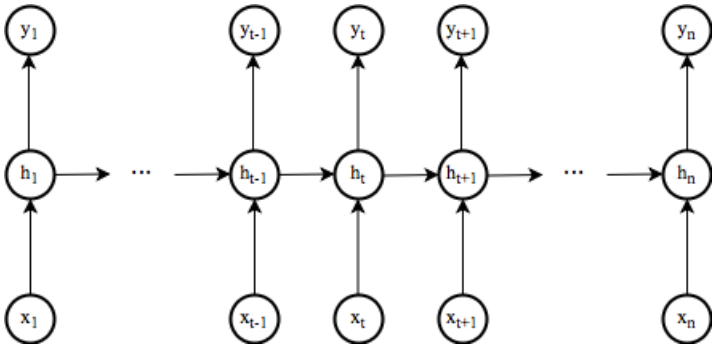


FIGURE 3.10 – Schéma d'un RNN avant.

b) RNN arrière :

Un RNN arrière fonctionne de la même façon qu'un RNN avant mais dans le sens inverse : de droite à gauche. Ce qui signifie que la prédiction se fait de la fin vers le début de la séquence (soit du futur vers le passé). Il consiste à récupérer l'information du noeud caché suivant pour calculer la sortie du noeud caché courant. La figure 3.11 illustre un RNN arrière.

$$h_t = \varphi(W_h h_{t+1} + W_x x_t + b_t) \tag{3.24}$$

$$y_t = \varphi(W h_t + b_y) \tag{3.25}$$

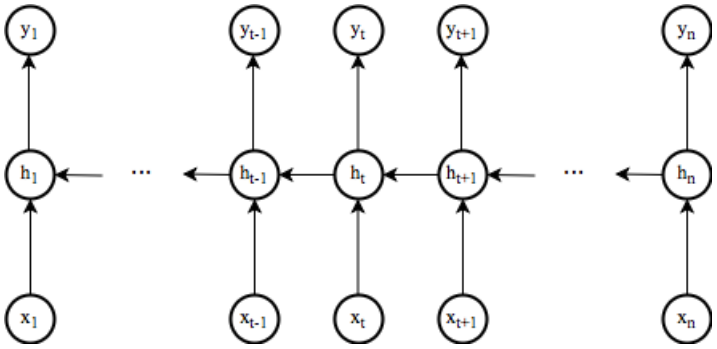


FIGURE 3.11 – Schéma d'un RNN arrière.

Les performances des RNN dépendent de la qualité de l'information à considérer dans un noeud courant. Cette information peut être étendue permettant l'accès aux informations passées $t-1$ et futures $t+1$. En pratique, ceci est réalisé avec un RNN bidirectionnel (Bi-RNN) [Schuster & Paliwal, 1997] composé de deux couches cachées : une couche pour une passe avant de x_1 à x_n, et l'autre couche pour une passe arrière qui lit l'entrée dans le sens inverse : de x_n à x_1.

L'avantage principal des RNN est leur capacité à utiliser des informations contextuelles du passé et/ou du futur. Cependant, un RNN peut rencontrer des difficultés pour capturer des dépendances à long terme. Les RNN ont notamment des difficultés à manipuler des séquences relativement longues

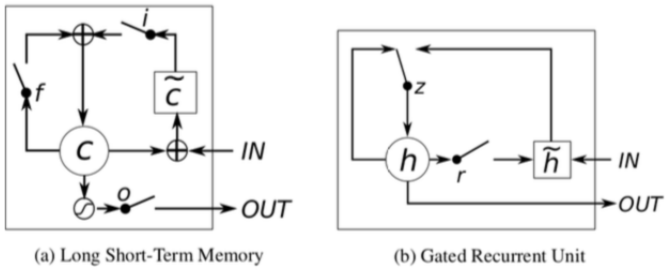
contenant plus de 10 unités récurrentes [Hochreiter *et al.*, 2001] . Ces difficultés sont dues à la variation exponentielle du coût L lors de l'apprentissage du RNN. L'influence d'une entrée sur les couches cachées augmente de façon exponentielle en fonction du nombre de connexions récurrentes. Autrement dit, le coût local à un instant t rétro-propagé dans le temps s'exprime de façon récursive en fonction des erreurs rétro-propagées aux instants passés [Bengio *et al.*, 1994]. En effet, avec cumul des calculs sur le long terme, le coût obtenu avec rétro-propagation du gradient décroît ou croît de façon exponentielle par rapport à la longueur de la séquence d'entrée. Ceci conduit respectivement aux problèmes de la dissipation du gradient (*vanishing gradient*) et l'explosion du gradient (*exploding gradient*). Il est important de noter que ces problèmes ne sont pas propres aux réseaux neuronaux récurrents. Ils apparaissent aussi dans des architectures neuronales profondes non récurrentes. Le problème de dissipation ou d'explosion du gradient s'accentue en fonction du nombre de couches.

Pour pallier à ce problème, la solution la plus utilisée dans la littérature consiste à remplacer l'unité récurrente classique par une unité récurrente avec des portes. Ces portes représentent des fonctions d'activation ajustant le flux d'information dans l'unité récurrente. On distingue deux types d'unités récurrentes avec des portes :

— Une unité récurrente à mémoire à court et long termes (Long-Short Term Memory LSTM) [Hochreiter & Schmidhuber, 1997] : elle est composée d'une mémoire (c) et de trois portes : d'entrée, de sortie et d'oubli. La porte d'entrée (i) choisit les informations pertinentes qui seront transmises à la mémoire. La porte de sortie (o) protège le réseau du contenu de sa mémoire. Et finalement, la porte d'oubli (f) permet à l'unité de mettre à zéro le contenu de sa mémoire.

— Une unité récurrente à portes (*Gates Recurrent unit* GRU) [Cho *et al.*, 2014] : elle est composé uniquement de deux portes : de réinitialisation et de modification. La porte de réinitialisation (r) décide d'ignorer ou pas l'état précédent de l'unité. La porte de modification (u) permet de décider si l'état caché (h) doit être mis à jour ou non.

La figure 3.12 illustre les deux unités LSTM et GRU.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

FIGURE 3.12 – Illustration des unités récurrentes de type LSTM et GRU. Dans LSTM (a) : c et $\widetilde{c}$ sont respectivement la mémoire et le nouveau contenu de la mémoire. Dans GRU (b) : h et $\widetilde{h}$ sont respectivement l'activation et l'activation candidate [Cho *et al.*, 2014].

L'architecture LSTM et ses variantes ont été largement appliquées avec succès pour plusieurs

tâches TALN. Parmi ces dernières, nous citons par exemple, la modélisation du langage [Mikolov *et al.*, 2011], la traduction automatique [Zeyer *et al.*, 2018], l'analyse d'opinions [Irsoy & Cardie, 2014; Tang *et al.*, 2015]. Nous présentons, dans la suite, les deux architectures LSTM et BiLSTM qui ont été largement utilisées pour la tâche d'analyse d'opinions.

3.3.2 Long-Short Term Memory

L'architecture LSTM est une catégorie de réseaux neuronaux récurrents dont l'architecture et la formulation mathématique générales sont identiques à celles présentés par la figure 3.8 et les formules 3.22 et 3.23.

La particularité des LSTM réside dans la façon avec laquelle le noeud caché est géré. Dans le cas des RNN simples, le traitement de la récurrence est assuré par une fonction d'activation φ qui est généralement une tangente hyperbolique *tanh*. Dans les LSTM, ce traitement est remplacé par une "unité à mémoire" schématisée dans la figure 3.12 qui remplace la couche cachée de la figure 3.8. L'unité LSTM est composée d'un noeud central, contenant l'état (ou la mémoire) interne de l'unité, et un nombre de trois portes. Ces portes permettent de gérer, d'une part, la mémorisation de l'information séquentielle (via les portes d'entrée et d'oubli) et, d'autre part, le rôle de l'état interne dans le calcul de la sortie (via la porte de sortie). Par exemple dans le cas d'une fermeture de la porte d'entrée, les nouveaux éléments sont moins pris en compte dans l'information de l'unité. Il est important de préciser que les les portes d'entrée et d'oubli sont calculées avant la mise à jour de la mémoire interne de l'unité.

Comme les RNN, les LSTM parcourent la séquence d'entrée dans un seul sens de x_1 à x_n. À chaque instant t, les portes consultent l'état précédent de l'unité (celui à l'instant $t-1$) pour calculer leurs nouvelles valeurs et mettre à jour la valeur de la mémoire c en appliquant les formules suivantes :

$$i_t = \varphi(W_i x_t + U_i h_{t-1} + b_i) \tag{3.26}$$

$$f_t = \varphi(W_f x_t + U_f h_{t-1} + b_f) \tag{3.27}$$

$$o_t = \varphi(W_o x_t + U_o h_{t-1} + b_o) \tag{3.28}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \varphi(W_c x_t + U_c h_{t-1} + b_c) \tag{3.29}$$

$$h_t = o_t \circ \varphi(c_t) \tag{3.30}$$

où : les matrices W et U reprèsentent respectivement les poids d'entrée et des noeuds récurrentes. i, f et o sont respectivement les portes d'entrée, d'oubli et de sortie, et c est le vecteur de mémoire interne ayant la même taille que le vecteur caché h.
Les LSTM ont prouvé leur efficacité dans diverses applications entre autre l'analyse d'opinions [Al-Azani & El-Alfy, 2017; Elnagar *et al.*, 2018b].

3.3.3 Long-Short Term Memory Bidirectionnels

Les réseaux de neurones de type LSTM présentés dans la section 3.3.2 sont simples. Ils utilisent uniquement le contexte précédent pour traiter l'élément courant dans une séquence d'entrée. Cependant, connaitre le contexte futur par rapport à un instant donné peut être aussi utile. Cette utilité se manifeste dans diverses applications du TALN, telles que l'étiquetage morpho-syntaxique, la traduction automatique, analyse de dépendance, *etc*.

Un LSTM bidirectionnel (BiLSTM) est un réseau récurrent bidirectionnel de type LSTM [Graves & Schmidhuber, 2005]. Il effectue les prédictions en tenant compte à la fois des informations passées et futures par rapport à un instant t dans la séquence d'entrée [Schuster & Paliwal, 1997]. La figure 3.13 illustre un BiLSTM qui est constitué d'une couche *LSTM* avant parcourant la séquence de x_1 à x_n et une autre couche *LSTM arrière* parcourant la séquence dans l'autre sens de x_n à x_1. Il s'agit donc d'une combinaison des LSTM avant et arrière qui sont entrainés conjointement.

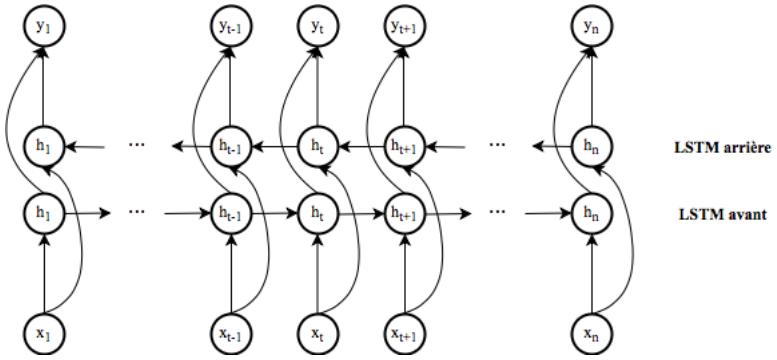


FIGURE 3.13 – Schéma d'un BiLSTM.

Les BiLSTM ont montré leur efficacité dans diverses applications entre autre l'analyse d'opinions [Heikal *et al.*, 2018].

3.4 Représentations continues de mots

Nous abordons, dans cette section, la deuxième partie de ce chapitre qui concerne l'étude de l'existant sur les représentations continues de mots. Le succès des réseaux de neurones présentés ci-dessus dépend beaucoup de la représentation des données en entrée. En analyse d'opinions, les données d'entrée sont des documents (une suite de mots). Les réseaux de neurones prennent alors en entrée des vecteurs de mots.

En TALN, une représentation de mots intuitive est la représentation *one hot* qui est un vecteur de la taille du vocabulaire considéré contenant une seule valeur non nulle pour la composante représentant le mot considéré. La taille de ce type de représentation est grande pour un vocabulaire grand. Ceci est vrai pour toutes les langues naturelles et la taille du vecteur *one hot* devient plus grande pour les langues riches morphologiquement (comme l'arabe). Ces représentations simples ne peuvent pas capter de relations syntaxiques et sémantiques entre les mots.

Les représentations continues de mots (*word embeddings*), dites aussi plongement de mots, représentent une méthode pour la représentation des mots dans un réseau de neurones. Ce paradigme repose sur l'hypothèse de distribution indiquant que les mots apparaissant dans des contextes similaires ont des significations semblables [Harris, 1954].

En se basant sur l'hypothèse de distribution, plusieurs méthodes ont été proposées pour construire des représentations vectorielles de mots capturant des liens entre les mots via une prise en compte des contexte de mots. Elles ont été utilisées avec succès dans plusieurs tâches du TALN telles que la reconnaissance des entités nommées, l'étiquetage morpho-syntaxique, le chunking, et en reconnaissance et compréhension de la parole.

Les plongements de mots sont des vecteurs denses à valeurs réelles et de dimension faible par rapport à la taille d'un vecteur *one hot*. Ils constituent une projection des mots du vocabulaire dans un espace vectoriel continu de faible dimension où chaque dimension représente une caractéristique latente du mot. Ces embeddings sont capables de préserver des similarités syntaxiques et sémantiques permettant ainsi de conserver les propriétés sémantiques et syntaxique de la langue [Mikolov *et al.*, 2013a]. Les mots proches sémantiquement ou syntaxiquement ont des vecteurs proches dans l'espace d'embeddings.

Pour avoir une idée plus claire sur les embeddings de mots, il est possible de procéder par des méthodes de visualisation. Par exemple, la technique *t-SNE* (*t-Distributed Stochastic Neighbor Embedding*) est une technique non linéaire de réduction de dimensions permettant de projeter des vecteurs de grande dimension dans un espace à deux ou trois dimensions [Turian *et al.*, 2010]. Les embeddings de mots constituent un nuage de points dans le nouvel espace. Cette technique permet de visualiser des îlots de mots partageant certaines caractéristiques. La figure 3.14 illustre une partie de l'espace $2D$ contenant des embeddings de mots, composé d'un îlot de mots portant une information numérique et un îlot de mots portant une information sur l'emploi.

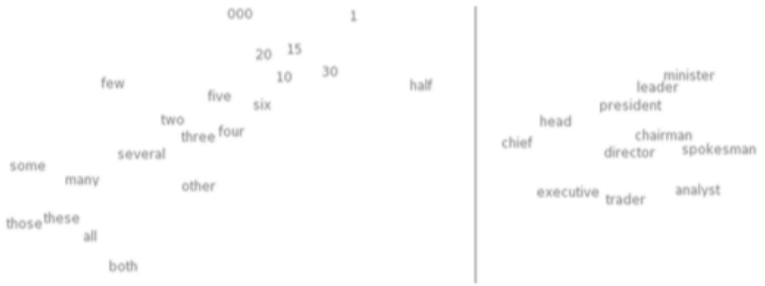


FIGURE 3.14 – Visualisation des plongements de mots dans un espace à deux dimensions [Turian *et al.*, 2010].

3.4.1 Méthodes de construction

Plusieurs approches neuronales ont été proposées pour construire des embeddings de mots. Nous présentons, dans la suite, les méthodes les plus utilisées, comme word2vec [Mikolov *et al.*, 2013b],

FastText [Bojanowski *et al.*, 2016], Elmo [Peters *et al.*, 2018] et BERT [Devlin *et al.*, 2018].

3.4.1.1 Word2vec

Word2vec est une méthode populaire de construction d'embeddings de mots. Elle a été introduite par [Mikolov *et al.*, 2013a,b]. Deux variantes de word2vec ont été proposées pour l'apprentissage des embeddings de mots : Skip-gram et CBOW. La figure 3.15 illustre les deux variantes word2vec. Chacune de ces architectures est composée de trois couches : une couche d'entrée, une couche cachée et une couche de sortie. La couche de sortie est composée de neurones avec une fonction d'activation *softmax*.

(A) Skip-gram. (B) CBOW

FIGURE 3.15 – Architectures de Skip-gram (A) et CBOW (B).

a) CBOW

L'architecture CBOW (*Continuous Bag Of Words*) permet de prédire un mot ω_i en fonction de son contexte c, selon une fenêtre à gauche et à droite du ω_i. Ce modèle suppose que l'ordre des mots du contexte n'a pas d'influence sur la projection. La couche de projection est partagée par tous les mots. L'apprentissage des embeddings de mots avec l'architecture CBOW consiste à prédire un mot en fonction de son contexte. Ceci consiste à calculer le vecteur résultant de la somme des embeddings de mots constituant le contexte, puis appliquer sur le vecteur résultant un classifieur log-linéaire pour prédire le mot cible. CBOW cherche à maximiser la vraisemblance selon la formule 3.31.

$$\frac{1}{|v|}\sum_{i=1}^{|v|} logp(\omega_i|\omega_{i-c},\ldots,\omega_{i-1},\omega_{i+1},\ldots,\omega_{i+c}) \tag{3.31}$$

où $|v|$ est la taille du vocabulaire du corpus d'apprentissage, c est la taille de la fenêtre de contexte et ω_i est le mot cible.

b) Skip-gram

L'architecture Skip-gram est également un réseau de neurones à trois couches et log-linéaire. Il permet, contrairement au CBOW, de prédire une fenêtre de contexte c sachant le mot ω_i au centre du contexte. Le mot ω_i représente l'entrée du réseau, et les mots du contexte forment la sortie. Il consite donc à prédire, pour un mot donné, son contexte. Par conséquent, l'embedding d'un mot quelconque sera proche des embeddings de mots se trouvant dans le même contexte. L'architecture Skip-gram maximise la vraisemblance selon la formule 3.32.

$$\frac{1}{|v|}\sum_{i=1}^{|v|}\sum_{j=i-c,j\neq i}^{i+c} logp(\omega_i|\omega_{i+j}) \tag{3.32}$$

où $|v|$ est la taille du vocabulaire du corpus d'apprentissage, c est la taille de la fenêtre de contexte et ω_i est le mot d'entrée.

Pour des raisons de complexité algorithmique, [Mikolov *et al.*, 2013b] ont proposé deux alternatives : l'échantillonnage négatif (*negative sampling*) et le softmax hiérarchique présentées ci-dessous :

— **Échantillonnage négatif** : il est basé sur le concept d'estimation contrastive de bruit (*noise contrastive estimation*) [Gutmann & Hyvärinen, 2012] supposant qu'un bon modèle devrait distinguer les échantillons négatifs des échantillons positifs par le biais d'une régression logistique. C'est-à-dire, au lieu de changer tous les poids à chaque fois en prenant en compte tous les mots du vocabulaire, on sélectionne seulement k échantillons négatifs pour mettre à jour les poids. (k étant la valeur de l'hyper-paramètre *negative sampling*). [Mikolov *et al.*, 2013b] ont montré que une valeur entière de k entre 5 et 20 est utile pour des petits corpus d'apprentissage, tandis que pour des grands corpus, la valeur de k peut être petit (entre 2 et 5).

— **Softmax hiérarchique** : il représente une approximation efficace du softmax. Au lieu d'évaluer les poids W du réseau de neurones pour obtenir la distribution de probabilité, il est uniquement nécessaire d'évaluer $\log_2(W)$ poids. En pratique, il utilise un arbre de Huffman pour réduire le temps de calcul. En effet, la variable logicielle hiérarchique utilise une représentation sous forme d'arborescence binaire de la couche en sortie avec les mots comme feuilles et, pour chaque nœud, représente explicitement les probabilités relatives de ses nœuds enfants.

Les deux alternatives d'échantillonnage négatif et de softmax hiérarchique permettent d'accélérer l'apprentissage des embeddings. Le temps d'apprentissage des embeddings raisonnable permet au modèle word2vec d'apprendre sur des corpus plus grands, un plus grand nombre de données d'apprentissage impliquant de meilleures représentations. Le Softmax hiérarchique fonctionne mieux pour les mots rares alors que l'échantillonnage négatif fonctionne mieux pour les mots fréquents [Mikolov *et al.*, 2013b].

Word2vec est capable de capturer des régularités sémantiques et syntaxiques de la langue considérée. Les angles entre certains mots sont corrélés aux relations qui relient ces mots. Parmi ces relations, nous citons par exemple, les relations féminin/masculin, singulier/pluriel, pays/capitale [Mikolov *et al.*, 2013b]. Il est possible d'exploiter ces relations avec des opérations arithmétiques simples

sur les vecteurs de mots.

3.4.1.2 FastText

FastText (FT) est une méthode de construction d'embeddings [Bojanowski *et al.*, 2016]. C'est une extension de word2vec [Mikolov *et al.*, 2013b], qui prend en compte des informations de type n-grammes. Word2vec considère les mots comme des unités lexicales atomiques. Les mots sont considérés comme insécables : chaque mot représente l'unité la plus petite. Par opposition à word2vec, FastText considère les mots comme des sacs de n-grammes de caractères.
FastText réduit la taille de l'unité atomique en considérant des n-grammes. Chaque n-gramme constitue un sous-mot. FastText associe des vecteurs aux n-grammes de caractères, et les mots sont alors représentés par la somme de ces vecteurs. Ainsi, fastText est capable d'extraire plus de relations sémantiques entre des mots partageant le caractère commun n-grammes. Il permet également d'obtenir des plongements pour des mots rares jamais vus en faisant la somme de ses vecteurs à n-grammes de caractères connus.

Pour expliquer le principe de fonctionnement de FT, nous prenons comme exemple le mot suivant "bonheur". FastText ajoute des symboles de début (<) et de fin (>) au mot pour obtenir le mot <bonheur>. Si n = 2, alors les sous-mots sont : {<b, bo, on, nh, he, eu, ur, r> }.
L'idée est d'apprendre, en plus des embeddings de mots, des représentations pour les n-grammes de caractère. Un n-gramme de caractères est un ensemble de n caractères consécutifs. Ces n-grammes de caractères sont appris de manière à ce que la somme de tous les embeddings des n-grammes de caractères formant un mot soit égale à l'embedding de ce mot. L'avantage de cette approche est qu'aucune séquence de caractères "utile" spécifique n'est définie à l'avance et que des mots inconnus/rares peuvent toujours être représentés par la somme des représentations de n-grammes de caractères, même si aucune représentation de mot n'est disponible. De plus, dans un contexte bruyant où certains caractères sont manquants ou sont ajoutés à un mot, la signification originale peut toujours être construite.

Diverses études montrent que le calcul des embeddings de mots sur des n-grammes de caractères à l'aide de FastText donne des résultats proches à l'utilisation de word2vec au niveau des mots [Joulin *et al.*, 2017; Cliche, 2017; Schmitt *et al.*, 2018; Pylieva *et al.*, 2018]. De plus, FastText affiche des résultats comparables, mais nettement moins de temps d'entraînement.

3.4.1.3 Autres Méthodes : Elmo et Bert

Des nouvelles méthodes de construction d'embeddings sont récemment apparues. Nous citons principalement : *Elmo* [Peters *et al.*, 2018] et *BERT* [Grave *et al.*, 2018]. Ces embeddings contextuels sont capables de gérer à la fois les contextes linguistiques et la syntaxe/sémantique des mots.
Les embeddings *Elmo* et *BERT* ne seront pas intoduits dans mes travaux actuels et feront partie de leurs perspectives. Dans cette thèse, nous choisissons de manipuler des embeddings de type word2vec et FastText.

3.4.2 Techniques d'évaluation

Les techniques d'évaluation des embeddings appartiennent à deux catégories : intrinsèque et extrinsèque. D'une part, les méthodes d'évaluation intrinsèque [Baroni *et al.*, 2014; Schnabel *et al.*, 2015] consistent à quantifier directement diverses régularités linguistiques dans l'espace d'embeddings. Les analogies syntaxiques et sémantiques [Mikolov *et al.*, 2013b; Nayak *et al.*, 2016] sont les méthodes intrinsèques les plus utilisées. D'autre part, les méthodes d'évaluation extrinsèque évaluent la qualité des embeddings pour d'autres tâches du TALN telles que l'étiquetage morphosyntaxique, la reconnaissance d'entités nommées, l'analyse d'opinions, *etc*.

3.4.2.1 Méthodes d'évaluation intrinsèque

Dans le cadre de l'évaluation intrinsèque, les embeddings de mots sont jugés sur la base des relations entre les mots par des humains. Des ensembles de mots créés manuellement sont souvent utilisés pour obtenir des évaluations humaines, puis ces évaluations sont comparées aux embeddings de mots (cette méthode de collecte des jugements est appelée évaluation intrinsèque absolue). La collecte des évaluations peut être réalisée par un ensemble limité de candidats (jugements recueillis en interne) ou sur des plateformes web de *crowdsourcing* comme Mechanical Turk (jugements collectés par crowdsourcing) [Liza & Grzes, 2016].

La méthode de l'analogie des mots est la méthode d'évaluation d'embeddings de mots la plus populaire. Elle permet de vérifier des régularités linguistiques et des cohérences sémantiques des mots dans une langue donnée. Elle est basée sur l'idée que les opérations arithmétiques dans un espace d'embeddings pourraient être prédites par l'homme. En effet, étant donné un ensemble de trois mots, a, b et c, la tâche consiste à identifier un mot d tel que la relation c :d est identique à la relation a :b [Turian *et al.*, 2010; Pereira *et al.*, 2016; Baroni *et al.*, 2014]. Par exemple, si les mots a = Athens, b = Greece, c = Oslo, alors le mot cible d est Norway puisque la relation a :b est capitale :pays, il faut donc trouver la capitale de quel pays est Oslo (voir la figure 3.16).

3.4.2.2 Méthodes d'évaluation extrinsèque

Dans le cadre de l'évaluation extrinsèque, les embeddings de mots sont évalués en se basant sur les performances des systèmes qui les ont utilisés pour différentes tâhces. Dans la suite, nous citons quelques tâches TALN pour lesquelles les embeddings de mots ont été utilisés.

— **Chunking du groupe nominal** : Cette tâche consiste à identifier les groupes nominaux et préciser leurs limites dans la phrase. Ceci revient, en pratique, à marquer tous les groupes nominaux sous forme [noms] + [mots attaché] [Turian *et al.*, 2010; Collobert *et al.*, 2011a; Schnabel *et al.*, 2015].

— **Reconnaissance des entités nommées** : Elle consiste à identifier les types d'entités nommées (noms d'organisations, de personnes, de marques, etc.) dans la phrase et leurs limites [Turian *et al.*, 2010; Collobert *et al.*, 2011a].

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

FIGURE 3.16 – Exemples d'analogie sémantique et syntaxique [Mikolov *et al.*, 2013a].

— **Analyse d'opinions** : Étant un cas particulier de problème de classification de texte, l'analyse d'opinions consiste à identifier pour un énoncé textuel donné sa polarité. Plusieurs travaux ont utilisé les embeddings de mots dans l'analyse d'opinions [Kim, 2014b; Tai *et al.*, 2015; Dai & Le, 2015; Tang *et al.*, 2016].

Dans le cadre du traitement automatique de la langue arabe, les embeddings de mots sont aussi testés pour différentes tâches telles que la traduction automatique [Shapiro & Duh, 2018; Lachraf *et al.*, 2019], l'analyse d'opinions [Dahou *et al.*, 2016; Soliman *et al.*, 2017; Fouad *et al.*, 2019] et la recherche d'information [El Mahdaouy *et al.*, 2018]. La majorité des travaux du TALN en arabe ont montré l'utilité des embeddings de mots.

3.5 Réseaux de neurones pour l'AO en arabe : état de l'art

Nous présentons, dans cette section, l'état de l'art des travaux réalisés en AOA utilisant des réseaux de neurones artificiels. Nous commençons par présenter les méthodes neuronales utilisées et les prétraitements appliqués, et nous finissons par une synthèse. Le tableau 3.1 résume ces travaux en les classant par ordre d'apparition chronologique.

3.5.1 Méthodes neuronales

Les premiers réseaux de neurones artificiels utilisés en AOA sont : réseau de neurones profond (DNN), combinaison de réseaux de type DBN (*deep belief network*) et DAE (*deep autoencoder*), et auto-encodeur récursif (RAE) pour la classification de phrases en ASM [Al Sallab *et al.*, 2015]. Le réseau DNN a été appliqué aussi pour l'AD [Abdelhade *et al.*, 2017]. Étant le plus performant, le

Travail	Niveau	Méthodes	Corpus	Résultats
[Al Sallab *et al.*, 2015]	phrase (ASM)	DNN, DBN, DBN+DAE, RAE	Arabic Tree-Bank (ATB)	RAE=74.3% , DNN=39.5%, DBN= 41.3%, DBN+DAE=43.5%
[Dahou *et al.*, 2016]	phrase et document (ASM et AD)	CNN	LABR, HTL, MOV, PROD, RES, ASTD	LABR=89.6%, HTL=91.7%, MOV=80.7%, PROD=87.3%, RES=78.5%, ASTD=79.07%
[Al-Sallab *et al.*, 2017]	phrase (ASM)	RAE	ATB, QALB	ATB=77.3%, QALB=75.4%
[Baly *et al.*, 2017b]	phrase (ASM - stemme)	RNTN	ArSenTB	5classes=60.0%, 3classes=80.0%
[Baly *et al.*, 2017a]	phrase (ASM et AD - lemme)	RNTN	ASTD	4classes=58.5%
[Alayba *et al.*, 2017]	phrase (ASM et AD)	DNN, CNN	2026 tweets	CNN= 90%, DNN= 85%
[Abdelhade *et al.*, 2017]	phrase (ASM et AD)	DNN	tweets	DNN= 90.2%
[Al-Azani & El-Alfy, 2017]	phrase (ASM et AD)	CNN, LSTM, CNN+LSTM, LSTM-comb	ASTD	CNN=74.1%, LSTM= 80.1%, CNN+LSTM= 73.5%, LSTM-comb=81.6%
			ArTwitter	CNN= 83.2%, LSTM= 83.7%, CNN+LSTM=84.2%, LSTM-comb=87.3%
[Elnagar *et al.*, 2018b]	document (ASM et AD)	CNN, LSTM	BRAD	CNN= 89.61%, LSTM= 90.05%
[Alayba *et al.*, 2018b]	phrase (ASM et AD)	CNN + lexique	2026 tweets	CNN=92%
[Heikal *et al.*, 2018]	phrase	CNN, BiLSTM, vote	ASTD	CNN= 64.3%, BiLSTM=64.7%, vote=65.1%
[Alayba *et al.*, 2018a]	phrase	CNN+LSTM	ASTD, ArTwitter	ASTD= 88.1%, ArTwitter=76.4%
[Mohammed & Kora, 2019]	phrase	CNN, LSTM, CNN+LSTM	40k tweets	CNN= 75.7%, LSTM=81.3%, CNN+LSTM= 78.5%
[Dahou *et al.*, 2019]	phrase	DE-CNN	ASTD, ArTwitter	ASTD=81.1%, ArTwitter=91.8%

TABLE 3.1 – Travaux réalisés en AOA.

réseau RAE a été réutilisé avec une prise en compte de la spécificité morphologique de textes arabes (via la tokenisation) [Al-Sallab *et al.*, 2017].

Toute phrase en ASM obéit à des règles de construction, dites *grammaire*, et des structures particulières, dites *syntaxique*. La structure syntaxique d'une phrase est une hiérarchie de syntagmes, représentable par un arbre syntaxique dont les feuilles sont les mots qui composent la phrase et les noeuds sont les syntagmes. La structure syntaxique d'une phrase en arabe est ainsi régie par le mécanisme de grammaires de la langue arabe. Les grammaires peuvent être construites selon l'aspect de récursivité partant de la racine jusqu'aux feuilles.
La modélisation de la structure syntaxique d'une phrase paraît efficace pour l'AO en commençant par préciser les polarités des feuilles, puis étudier celles des noeuds intermédiaires, et finissant par déterminer la polarité de la racine (c'est-ç-dire la polarité de la phrase entière). Dans cette perspective, *Recursive Neural Tensor Networks* (RNTN), initialement conçu pour l'AO en anglais, est un réseau récursif qui a été appliqué pour l'AOA au niveau phrase [Baly *et al.*, 2017b,a]. Ce réseau nécessite un corpus spécifique appelé *sentiment treebank* qui représente une collection d'arbres syntaxiques annotés en polarité à tous les niveaux syntaxiques.

Par opposition aux réseaux récursifs qui se basent sur la structure hiérarchique de la phrase, les réseaux récurrents considèrent la phrase comme une séquence de mots. Ils se basent donc sur la structure linéaire de la phrase. Les réseaux récurrents ont été également appliqués pour l'AOA. Nous citons principalement les réseaux LSTM [Al-Azani & El-Alfy, 2017] et BiLSTM [Heikal *et al.*, 2018].

De plus, les réseaux convolutifs ont été largement utilisés [Dahou *et al.*, 2016; Alayba *et al.*, 2017, 2018b; Heikal *et al.*, 2018]. Le choix de l'architecture du CNN et ses différents hyper-paramètres est difficile. L'algorithme d'évolution différentielle *Differential Evolution* (DE) peut être utilisé pour rechercher automatiquement la configuration optimale, y compris l'architecture du réseau et ses hyper-paramètres [Dahou *et al.*, 2019].
Dans le travail de [Dahou *et al.*, 2019], cinq paramètres CNN sont recherchés par l'algorithme DE : taille de filtre de convolution, le nombre de filtres, le nombre de neurones dans la couche entièrement connectée (FC), le mode d'initialisation et le coefficient du dropout.

Afin de tirer profit des avantages de chaque réseau, des combinaisons de CNN et LSTM ont été également proposées : combinaison par vote [Heikal *et al.*, 2018] et combinaison séquentielle [Al-Azani & El-Alfy, 2017; Alayba *et al.*, 2018a].

La majorité des réseaux de neurones prend en entrée des embeddings de mots. Ces derniers peuvent être aléatoires ou pré-entrainés. Ils sont dans la majorité des travaux pré-entrainés avec des corpus. La nature et la taille de corpus intervient dans la qualité des embeddings pré-entrainés [Antoniak & Mimno, 2018; Qazanfari & Youssef, 2019]. La dimension d'embeddings intervient aussi dans leur qualité [Pierrejean & Tanguy, 2018]. En arabe, la majorité des embeddings sont de dimension 300 [Dahou *et al.*, 2016; Soliman *et al.*, 2017; Bojanowski *et al.*, 2017; Grave *et al.*, 2018].
Deux types d'entrainement des réseaux de neurones peuvent être distingués : non statique et statique [Kim, 2014a]. La version non statique consiste à mettre à jour les embeddings lors de l'apprentissage. Ce n'est pas le cas en version statique. D'après nos lectures, les réseaux non statiques sont plus

performants que ceux statiques [Dahou *et al.*, 2016; Al-Azani & El-Alfy, 2017].

3.5.2 Prétraitements

Toute méthode d'analyse de textes contient généralement une phase de prétraitements. Cette dernière consiste à nettoyer les corpus (les données textuelles) et les préparer à la tâche en question. Elle permet de réduire le bruit dans les données textuelles et garder les informations nécessaires et pertinentes à la classification. Le prétraitement de textes joue un rôle essentiel dans l'analyse d'opinions. En AOA, on distingue deux familles de prétraitements : i) prétraitement de base, ii) prétraitement spécifique à la langue arabe.

i) Prétraitements de base : ils regroupent toute sorte de prétraitement basique et simple pouvant être appliquée aux données d'opinions. Nous citons par exemple les prétraitements suivants :

- supprimer les urls, les hashtags, les mentions, les mots outils afin de réduire le bruit et ne garder que les mots utiles et pertinents à l'AO
- ne garder que deux occurrences des caractères identiques consécutifs. La répétition d'un caractère plusieurs fois reflète principalement l'intensité de la polarité et de l'émotion. Par exemple, les mots "looooove", "loooove" et "loove" traduisent l'intensité de la polarité positive. Afin de normaliser l'intensité, il est d'usage de se contenter de deux occurrences du caractère ("loove" pour notre exemple).
- identifier les émoticônes. Un émoticône représente une suite de caractères formant un visage stylisé et exprimant une émotion. Par exemple, l'émoticône :) symbolise un visage souriant, et l'émoticône :(symbolise un visage triste. Les émoticônes sont utilisés par les internautes afin d'exprimer des émotions, afficher des opinions et déclarer des polarités.

Les prétraitements de base sont appliqués dans la majorité des travaux d'analyse d'opinions indépendamment de la langue : anglais [Singh & Kumari, 2016; Wegrzyn-Wolska *et al.*, 2016], français [Abdaoui *et al.*, 2015; Benamara *et al.*, 2017] et arabe [Shoukry & Rafea, 2015; Alnawas & Arıcı, 2018].

ii) Prétraitements spécifiques à la langue arabe : Ils consistent à prendre en compte les caractéristiques de la langue arabe [Alnawas & Arıcı, 2018] (voir les sections 2.1 et 2.2 du chapitre 2). Deux types peuvent être distingués :

- Prétraitement spécifique simple : il traite principalement l'orthographe de la langue arabe. Nous citons, par exemple :
  - suppression des symboles non arabes
  - suppression des voyelles courtes et des signes diacritiques
  - normalisation des caractères أ, إ, آ en ا
  - normalisation des caractères ي, ى en ي
  - normalisation du caractère ة en ه

— ne garder qu'une occurence du caractère *tatweel* puisque ce dernier, spécifique à la langue arabe [Jamal *et al.*, 2006], reflète l'intensité de la polarité ou de l'émotion. Par exemple, les mots جميـــــل, جميـل (exprimant une polarité très positive) sont normalisés en جميل

— Prétraitement spécifique avancé : il prend en compte les spécificités de la langue arabe afin de réduire la taille du vocabulaire. Nous citons par exemple, la segmentation [Al-Sallab *et al.*, 2017], la lemmatisation [Baly *et al.*, 2017a] et le stemming [Baly *et al.*, 2017b].

Il nous importe à signaler que ces prétraitements ne sont pas spécifiques aux méthodes neuronales. Ils ont été également utilisés avec les autres méthodes symboliques, numériques et hybrides [Abdul-Mageed *et al.*, 2011; Mountassir *et al.*, 2013a].

3.5.3 Synthèse

La recherche en analyse d'opinions en arabe est relativement récente par rapport à celle en anglais. L'objectif des systèmes d'AOA consiste en la détermination de la polarité. Plusieurs niveaux de granularité de la polarité peuvent être distingués :

— positive et négative
— positive, neutre et négative
— très positive, positive, neutre, négative, très négative
— positive, négative, neutre et mixte

La majorité des travaux à base de réseaux de neurones se situe dans le cadre d'une classification binaire [Al Sallab *et al.*, 2015; Dahou *et al.*, 2016; Al-Sallab *et al.*, 2017; Alayba *et al.*, 2017, 2018b; Dahou *et al.*, 2019]. Quelques travaux ont été réalisés pour une classification ternaire [Baly *et al.*, 2017b] et quinaire [Baly *et al.*, 2017b]. Une classification quaternaire a été également effectuée considérant les classes positive, négative, objective (ou neutre) et mixte [Baly *et al.*, 2017a].

Les réseaux de neurones sont plus performants que les classifieurs basiques (SVM, NB, KNN) [Dahou *et al.*, 2016; Elnagar *et al.*, 2018b]. Les réseaux de neurones récursifs ont montré du succès pour l'analyse d'opinions au niveau phrase en ASM [Baly *et al.*, 2017b,a] Les réseaux convolutifs et récurrents ont également prouvé leur efficacité pour l'analyse d'opinions au niveau phrase et document [Dahou *et al.*, 2016; Elnagar *et al.*, 2018b; Heikal *et al.*, 2018; Dahou *et al.*, 2019]. Ces réseaux sont basés sur des embeddings de mots [Dahou *et al.*, 2016] et des embeddings de caractères [Omara *et al.*, 2018].

Les ressources (corpus et lexique polarisé) de l'AOA ne sont pas aussi nombreux que l'AO en anglais. Les premiers corpus sont petits (entre 500 et 1000 documents) et ils sont annotés en polarité par des humains. L'annotation manuelle nécessite des annotateur humains et elle demande beaucoup de temps. Vu la difficulté d'annotation manuelle de corpus en polarité, une méthode automatique de construction de corpus est apparue : la polarité n'est pas déterminée par un annotateur humain mais par la personne qui a laissé le commentaire. La méthode automatique de contruction de corpus permet de mettre en disponibilité des corpus suffisamment grands (des disaines de milliers de documents). Cependant, la majorité de ces corpus sont déséquilibrés (pas de répartition uniforme des polarités).

Ce déséquilibre dans le corpus peut détériorer les performances du modèle par rapport aux classes minoritaires avec les classifieurs basiques ou les réseaux de neurones profonds [Ando & Huang, 2017].
En plus de la taille du corpus et sa répartition de polarité, la structure d'un document est complexe, il peut être composé d'un seul mot, un groupe de mots, une phrase ou plusieurs phrases.
Le problème de la négation est soulevé dans la majorité des travaux en AOA [Alayba *et al.*, 2017; Abdelhade *et al.*, 2017; Alayba *et al.*, 2018b].

3.6 Conclusion

Dans ce chapitre, nous avons présenté les modèles de classification neuronaux. Nous avons abordé, dans un premier temps, une description théorique des réseaux de neurones. Il existe de nombreuses architectures neuronales. Nous avons particulièrement détaillé le fonctionnement de base des réseaux de neurones convolutifs et des réseaux récurrents de type LSTM et BiLSTM permettant d'obtenir de bonnes performances. Cette restriction de présentation explique le choix des architectures neuronales utilisées dans cette thèse. Nous avons décrit, dans un deuxième temps, les différentes méthodes de construction d'embeddings et leurs techniques d'évaluation. À la fin, nous avons établi l'état de l'art des méthodes neuronales pour l'analyse d'opinions en arabe.

Nous allons présenter, dans le reste de cette thèse, notre approche neuronale pour l'analyse d'opinions en arabe.

Chapitre 4

Systèmes neuronaux pour l'analyse d'opinions en arabe : une étude préliminaire

Sommaire

4.1 Systèmes d'analyse d'opinions 82
4.1.1 Architecture neuronale 82
4.1.2 Embeddings de mots utilisés 83
4.2 Cadre expérimental 85
4.2.1 Corpus 85
4.2.2 Lexique polarisé 87
4.2.3 Métriques d'évaluation 89
4.3 Premières expériences 89
4.3.1 Impact des pré-traitements du corpus 90
4.3.2 Premières évaluations du CNN 91
4.3.3 Représentation des documents 94
4.4 Embeddings de mots : étude qualitative 97
4.4.1 Couverture 97
4.4.2 Étude du voisinage 99
4.5 Conclusion 100

Dans ce chapitre, nous proposons un protocole expérimental pour l'analyse d'opinions en arabe. Nous présentons, tout d'abord, l'architecture neuronale utilisée pour implémenter nos systèmes d'AOA. Nous proposons également une étude qualitative sur les embeddings de mots de type word2vec [Mikolov *et al.*, 2013b] et FastText [Bojanowski *et al.*, 2016].

Ce chapitre est organisé comme suit. Nous détaillons dans la section 4.1 les systèmes d'analyse d'opinions utilisés : l'architecture neuronale et les embeddings de mots utilisés comme entrée. Nous

présentons ensuite le cadre expérimental dans la section 4.2. Puis, nous reportons les performances de nos systèmes dans la section 4.3. Enfin, nous menons dans la section 4.4 une étude qualitative des embeddings de mots : entrée de l'architecture neuronale.

4.1 Systèmes d'analyse d'opinions

Nos systèmes d'AO sont fondés sur l'architecture neuronale basée sur les réseaux de neurones convolutifs (CNN). Nous détaillons dans la suite notre architecture et présentons les embeddings de mots utilisés comme entrée au réseau convolutif.

4.1.1 Architecture neuronale

Selon notre étude de l'état de l'art, l'architecture neuronale de type CNN est très utilisée pour l'analyse d'opinions. Cette architecture a fait ses preuves pour l'analyse d'opinions en anglais [Kim, 2014a]. Nous décidons alors de tester cette architecture pour l'AOA. Notre système se base sur un réseau de neurones convolutif. L'architecture du CNN est similaire à celle proposée dans [Kim, 2014a] pour l'analyse d'opinions en anglais (voir figure 4.1). Cette dernière a été testée par [Dahou *et al.*, 2016] pour l'AOA et a donné de bonnes performances. Nous choisissions alors l'architecture de [Dahou *et al.*, 2016] pour mener notre étude préliminaire.

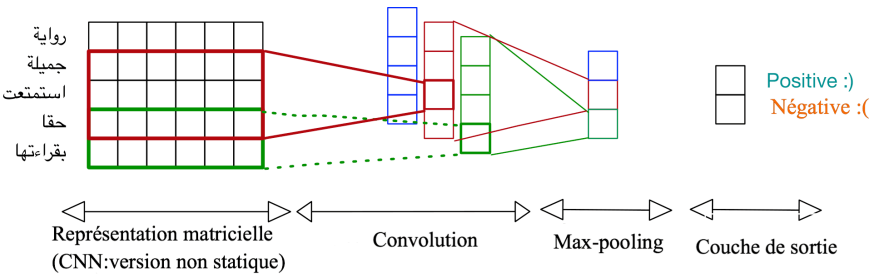


FIGURE 4.1 – Architecture CNN de [Kim, 2014a]

L'architecture d'un CNN est généralement composée d'une séquence de couches de convolution et de pooling, suivie par des couches de neurones totalement connectées (voir la section 3.2 du chapitre 3).
Dans l'architecture du réseau CNN, l'entrée correspond à une séquence de mots d'un document (un commentaire). Le CNN prend en entrée une matrice d'*embeddings* de taille fixe $M \in \mathbb{R}^{n \times k}$, avec n le nombre de mots et k la dimension des embeddings. Il applique une convolution de filtres, dont la taille de la fenêtre est une des valeurs de l'ensemble $\{3,4,5\}$, pour extraire de nouveaux attributs à partir de la matrice d'*embeddings* M. Puis, un *max_pooling* est appliqué sur la sortie de chaque couche de convolution dans le but de conserver uniquement les attributs les plus pertinents qui sont concaténés au niveau d'une couche entièrement connectée. Enfin, ce CNN applique la fonction d'activation *softmax* à la couche de sortie pour générer la polarité du document fourni en entrée.

Pour ce réseau, un dropout de 0.5 est utilisé, c'est-à-dire que 50% des neurones sont désactivés à chaque itération. L'algorithme d'optimisation utilisé pour l'apprentissage est l'Adagrad [Duchi *et al.*, 2011]. C'est un algorithme d'optimisation basé sur le calcul de gradient [1] : il adapte le taux d'apprentissage en effectuant de plus petites mises-à-jour des paramètres. Il est adapté au traitement de données éparses. L'algorithme Adagrad améliore considérablement la robustesse de SGD [Dean *et al.*, 2012].

Nous récapitulons, dans le tableau 4.1, quelques hyper-paramètres du CNN. Nous avons gardé les mêmes valeurs que celles décrites dans [Dahou *et al.*, 2016] afin de reproduire leurs résultats. Nous considérons ce système comme une baseline dans cette thèse.

Hyper-paramètre	**Valeur**
Longueur du document n	882
Taille de filtres	$\{3,4,5\}$
Dropout	0.5
Version du CNN	non statique

TABLE 4.1 – Valeurs de quelques hyper-paramètres du CNN.

Notre système CNN est non statique, c'est-à-dire que les embeddings pré-entrainés sont mis à jour lors de l'apprentissage du CNN. Il nous importe à signaler que la valeur de la longueur du document n est fixe à déterminer. La longuer n a été fixée à 882 dans [Dahou *et al.*, 2016]. Nous détaillons, dans la section 4.3.3, notre protocole du calcul de longueur dans le cadre d'AO.

4.1.2 Embeddings de mots utilisés

Dans cette section, nous décrivons les ressources d'*embeddings* de mots en arabe disponibles gratuitement. Dans le cadre de notre étude préliminaire, nous utilisons ces embeddings comme entrée à l'architecture neuronale CNN. Deux ressources d'embeddings ont été entrainées avec le modèle word2vec [Mikolov *et al.*, 2013b] et deux autres ressources ont été entrainées avec fastText [Bojanowski *et al.*, 2016].

La première ressource est celle de [Dahou *et al.*, 2016]. Ils ont entrainé le modèle word2vec de type *Skip-gram* et *continuous bag of words* (CBOW) sur des pages web. Leurs expériences ont montré que CBOW est plus performant, ils l'ont donc mis à disposition.

La deuxième ressource [Soliman *et al.*, 2017] est plus riche : elle regroupe six modèles d'*embeddings* entrainés sur trois types de corpus différents : twitter, wikipédia et des pages web. [Soliman *et al.*, 2017] ont entrainé CBOW et Skip-gram sur les trois types de corpus, mettant ainsi à disposition six ensembles d'embeddings.

Par opposition aux deux premières ressources qui sont de type word2vec, les troisième et quatrième

1. http://ruder.io/optimizing-gradient-descent/index.html#fn10

ressources sont de type fastText. La troisième ressource regroupe deux ensembles d'embeddings entrainés avec Wikipedia [2] : un ensemble en arabe standard moderne (ASM) et un autre en arabe dialectal(AD) [Bojanowski *et al.*, 2017]. La quatrième ressource contient deux ensembles d'embeddings entrainés avec Common Crawl [3] en arabe standard moderne (ASM) et arabe dialectal(AD) [Grave *et al.*, 2018]. Ces quatre ensembles d'embeddings ont été entrainés en utilisant la version CBOW de fastText avec des n-grammes de caractères de longueur 5, une fenêtre de taille 5 et un échantillonnage négatif de valeur 10.

Nous disposons ainsi de 11 ensembles d'embeddings de mots : 7 ensembles entrainés avec word2vec et 4 ensembles entrainés avec fastText. Le vocabulaire de chaque ensemble d'embeddings word2vec est mixte, il regroupe à la fois des mots en ASM et AD. Quant aux ensembles d'embeddings FastText, deux ensembles sont en ASM et les deux autres sont en AD. Les tables 4.2 et 4.3 rapportent la taille des différents ensembles d'embeddings pre-entrainés respectivement avec word2vec et FastText.

Embeddings	**Word2vec**						
	[Dahou *et al.*, 2016]	[Soliman *et al.*, 2017]					
	Web	Twitter		Wikipédia		Web	
	CBOW	CBOW	Skip-gram	CBOW	Skip-gram	CBOW	Skip-gram
Nom	Dahou	twt-cbow	twt-sg	wiki-cbow	wiki-sg	www-cbow	www-sg
Taille	2 214 051	164 077	204 448	140 319	140 319	146 273	145 428

TABLE 4.2 – Taille de vocabulaire des différents ensembles d'embeddings pré-entrainés word2vec.

La taille de l'ensemble d'embeddings de [Dahou *et al.*, 2016] *Dahou_embed* est plus grande que celles des ensembles de [Soliman *et al.*, 2017] *Soliman_embed*. En effet, *Dahou_embed* contient, en plus des embeddings de mots, des embeddings de groupes de mots.
La taille des embeddings de type fastText varient entre 54k et 2 000k mots. Selon le principe de fastText basé sur la notion de *subword*, le modèle contient, en plus des embeddings de mots (dont le nombre est indiqué dans le tableau 4.3), des embeddings de n-grammes de caractères. Ces n-grammes de caractères constituent les *subwords*. En se basant sur les embeddings de n-grammes, on peut ainsi générer des embeddings pour des mots inconnus (ce sont des mots qui n'ont pas participé à l'entraînement de fastText). Ce qui signifie que le nombre de mots fictifs que nous pouvons construire est plus grand que le nombre de mots considérés (mentionné dans le tableau 4.3).

Nous utilisons les différents ensembles d'embeddings pour entrainer nos systèmes d'analyse d'opinions (présentés dans la section 4.1). Il est important de signaler que tous les embeddings disponibles sont de dimension 300. Ce qui permet de comparer la qualité des ensembles d'embeddings et la performance des systèmes neuronaux.

2. https://www.wikipedia.org
3. http://commoncrawl.org

Embeddings	FastText			
	[Bojanowski *et al.*, 2017]		[Grave *et al.*, 2018]	
	Wikipédia		Common Crawl	
	ASM	Dialecte	ASM	Dialecte
Nom	wiki.ar	wiki.arz	cc.ar	cc.arz
Taille	610 977	54 595	2 000 000	356 316

TABLE 4.3 – Taille de vocabulaire (nombre de mots) des différents ensembles d'embeddings pré-entrainés avec FastText.

4.2 Cadre expérimental

Notre protocole expérimental permet d'évaluer nos systèmes neuronaux. Nous avons utilisé le corpus LABR : le plus grand corpus disponible au commencement de la thèse qui est présenté dans la section 4.2.1. Nous présentons également un lexique polarisé qui nous servira plus tard dans le choix du protocole de représentations des documents dans la section 4.3.3. Nous précisons les métriques d'évaluation généralement utilisées pour calculer les performances de systèmes d'analyse d'opinions et comparer les systèmes entre eux.

4.2.1 Corpus

Nous présentons le corpus que nous utilisons pour la classification.

4.2.1.1 Composition du corpus

Pour évaluer nos systèmes, nous avons besoin d'un corpus d'opinions. Nous avons utilisé le corpus LABR [Nabil *et al.*, 2014] qui contient 63 257 critiques de livres composées d'un commentaire et d'une note associée (nombre d'étoiles) : chaque commentaire est noté avec une échelle de 1 à 5 étoiles. Le tableau 4.4 décrit la répartition des commentaires sur les étoiles de classement.

	Nombre d'étoiles					
Corpus	*	* *	* * *	* * * *	* * * * *	**Total**
Train_initial	2 337	4 197	9 841	15 216	19 015	50 606
Test	602	1 088	2 360	3 838	4 763	12 651

TABLE 4.4 – Distribution du corpus LABR sur l'échelle de 5 étoiles

Dans le cadre de telle notation de corpus, il est d'usage, dans la communauté scientifique, de considérer les cadres de classification suivants [Nabil *et al.*, 2014; Dahou *et al.*, 2016] :

— **Classification binaire** : les commentaires notés avec 1 ou 2 étoiles représentent la classe *négative* et les commentaires notés avec 4 ou 5 étoiles constituent la classe *positive*. Les commentaires notés avec 3 étoiles sont écartés.

— **Classification ternaire** : les commentaires notés avec 1 ou 2 étoiles représentent la classe *négative*, les commentaires notés avec 3 étoiles forment la classe *neutre*, et les commentaires notés avec 4 ou 5 étoiles constituent la classe *positive*.
— **Classification quinaire** : les commentaires notés avec 1 étoile représentent la classe *très négative*, ceux notés avec 2 étoiles constituent la classe *négative*, les commentaires notés avec 3 étoiles forment la classe *neutre*, ceux notés avec 4 étoiles représentent la classe *positive* et les commentaires notés avec 5 étoiles constituent la classe *très positive*.

Le corpus LABR contient plus de trois millions de mots sur un vocabulaire de taille 324k. Pour mieux comprendre la distribution des mots, il est intéressant de connaitre les quelques statistiques suivantes : le nombre d'occurrences du mot le plus fréquent est de 76 855 quand il est à 319 pour le 1000^{ime} mot le plus fréquent. Si on considère qu'un mot fréquent est un mot qui apparait plus de 5 fois dans le corpus, alors le vocabulaire de mots fréquents représente 13% du vocabulaire initial et il couvre 86.5% du corpus.

4.2.1.2 Préparation des ensembles Train, Dev et Test

Classiquement, un corpus se décompose en trois ensembles Train, Dev et Test :

— L'ensemble Train est utilisé pour entraîner le système de classification permettant la construction d'un modèle de classification. Le modèle consiste en l'ensemble condensé des informations pertinentes retenues pour la détermination des classes.
— L'ensemble Dev est utilisé pour ajuster des paramètres du système en vue de son optimisation. Les ajustements s'effectuent en fonction des résultats de classification obtenus sur l'ensemble Dev afin de rendre le modèle plus robuste. La meilleure configuration est choisie en fonction des meilleurs résultats observés sur l'ensemble Dev.
— L'ensemble Test représente une portion de corpus que le système n'a jamais vu au cours de son apprentissage et de son ajustement. Il est utilisé pour obtenir les performances finales du système une fois l'apprentissage fini. Les performances obtenues sur le corpus Test représentent l'évaluation du système. Elles estiment sa capacité de généralisation, c'est-à-dire de s'adapter à de nouvelles données jamais rencontrées.

Comme décrit dans le tableau 4.4, le corpus LABR contient un ensemble d'apprentissage et un autre de test. Mais, il ne dispose pas d'ensemble de validation. Nous avons crée un corpus de validation *Dev* en considérant 10% du corpus d'apprentissage *Train_initial* afin d'entrainer proprement nos systèmes. Les tableaux 4.5, 4.6 et 4.7 présentent les répartitions de LABR dans les 3 cadres de classification binaire, ternaire et quinaire respectivement.

En ce qui concerne la distribution des documents du corpus *Dev*, nous avons considéré la même distribution du corpus *Test* correspondant. Par exemple, dans le cadre d'une classification binaire, nous avons, tout d'abord, construit les ensembles *Train_initial* et *Test* à partir du corpus LABR. Ensuite, nous avons calculé la proportion de chaque classe dans le corpus *Test* : 83% pour la classe *positive* et 17% pour la classe *négative*. Puis, nous avons construit le corpus *Dev* à partir du corpus

Train_initial de façon à ce que la proportion du *Dev* soit égale à celle du *Test* : c'est-à-dire les proportions des classes *positive* et *négative* dans le corpus *Dev* sont respectivement 83% et 17%. Les documents restants de *Train_initial* constituent le corpus *Train*. Nous avons effectué le même processus afin de construire les corpus *Train*, Dev et *Test* pour les classifications ternaire et quinaire.

	Négative	**Positive**	**Total**
Train	5 840	30 848	36 688
Dev	694	3 383	4 077
Test	1 690	8 601	10 291

TABLE 4.5 – Répartition de LABR dans le cadre de la classification binaire

	Négative	**Neutre**	**Positive**	**Total**
Train	5 825	8 879	30 841	45 545
Dev	709	962	3 390	5 061
Test	1 690	2 360	8 806	12 651

TABLE 4.6 – Répartition de LABR dans le cadre de la classification ternaire

	Très négative	**Négative**	**Neutre**	**Positive**	**Très positive**	**Total**
Train	2 083	3 723	8 879	13 647	18 833	45 545
Dev	254	456	962	1 569	1 820	5 061
Test	602	1 088	2 360	3 838	4 763	12 651

TABLE 4.7 – Répartition de LABR dans le cadre de la classification quinaire

4.2.1.3 Pré-traitement du corpus

Le pré-traitement a pour objectif le nettoyage du corpus. Dans cette perspective, les opérations suivantes ont été effectuées :

- Supprimer les urls, les mentions, les hashtags, les nombres, les mots non arabes, les signes diacritiques
- ne garder qu'une occurence du caractère *tatweel*
- Ne garder que 2 occurrences des caractères identiques consécutifs
- Normaliser des caractères أ, إ, آ en ا
- Isoler les émoticônes

4.2.2 Lexique polarisé

Dans cette thèse, nous avons collecté tous les lexiques de sentiment disponibles à notre connaissance [Badaro *et al.*, 2014b; ElSahar & El-Beltagy, 2015; Saif M. Mohammad, 2016; Al-Moslmi

et al., 2018]. Cela représente un ensemble de 15 lexiques construits avec différentes méthodes.
La première méthode consiste à traduire automatiquement les lexiques anglais. En effet, les ressources traduites [Saif M. Mohammad, 2016] sont obtenues en traduisant les quatre lexiques anglais suivants : MPQA [Wilson *et al.*, 2005], S140 [Kiritchenko *et al.*, 2014], NRC [Mohammad & Turney, 2010; Mohammad & Yang, 2011; Mohammad *et al.*, 2013] et le lexique de Bing liu [Hu & Liu, 2004b].
La deuxième méthode est basée sur des informations mutuelles ponctuelles (PMI) entre les mots et deux étiquettes (positive et négative). En effet, la polarité d'un mot [Mohammad & Turney, 2013] représente la différence entre les scores PMI.

Les lexiques utilisés ont des tailles et des structures différentes. En fait, chaque mot est décrit avec des descripteurs différents. Ces derniers varient d'un lexique à l'autre. [Al-Moslmi *et al.*, 2018] a construit un lexique de 3 880 synsets positifs et négatifs annotés avec des étiquettes morpho-syntaxiques, la polarité, les scores polarité et une liste de synonymes.
Par conséquent, notre lexique *ArSentLex* regroupe plusieurs types d'annotations des mots en polarité, telles que (i) la traduction de lexiques polarisés existants et (ii) la méthode PMI.

ArSentLex est défini comme un tuple défini (ω, *pos*, *ps*, *ns*, *p*), où : ω est un mot, *pos* son étiquette morpho-syntaxique, *ps* son score de positivité, *ns* son score de négativité et *p* sa polarité (positive ou négative). Autrement dit, chaque ω est décrit par quatre descripteurs *pos*, *ps*, *ns* et *p* . Nous avons choisi de garder les descripteurs les plus pertinents à la tâche d'analyse d'opinions. Nous avons supposé que la polarité d'un mot peut être positive ou négative. Néanmoins, étant conscient de la bi-polarité de certains mots, les scores de positivité et de négativité sont utiles comme descripteurs. La construction de tel lexique suppose généralement que les mots qui n'y figurent pas sont implicitement neutres.
Pour la concaténation des 15 lexiques, nous avons effectué les étapes suivantes :

— Lorsque le descripteur de polarité *p* n'est pas présent dans un lexique, nous associons la polarité en fonction des scores de positivité et de négativité. Cela signifie que la polarité positive est attribuée à un mot lorsque le score de positivité est supérieur à celui de négativité et inversement.
— Lorsqu'un mot ω est décrit par une liste de synonymes comme dans le lexique de [Al-Moslmi *et al.*, 2018], nous ajoutons autant de lignes qu'il y a de synonymes. Pour chaque synonyme, nous attribuons les mêmes descripteurs que ceux de ω.

Ainsi, *ArSentLex* contient 51 968 mots positifs et 45 638 mots négatifs. Parmi lesquels, 11 088 mots sont à la fois négatifs et positifs. Le tableau 4.8 rapporte quelques exemples de mots positifs, négatifs et, positifs et négatifs à la fois dans *ArSentLex*.

Mots positifs	Mots négatifs	Mots positifs et négatifs
يضحكني /yDHkny/ (me fait rire)	فوضوي /fwDwyt/ (désordonné)	منتظر /mntZr/ (attendu)
السعادة /AlsEdt/ (la joie)	اجرامي /AjrAmy/ (criminel)	تمازج /tmAzj/ (un mélange)
عيد /Eyd/ (une fête)	اشكال /A$kAl/ (une problématique)	المثالية /Almvlyt/ (l'idéalisme)

TABLE 4.8 – Exemples de mots polarisés dans *ArSentLex*.

4.2.3 Métriques d'évaluation

Afin de mesurer la performance d'un système d'analyse d'opinions, il est important de choisir la mesure adéquate à la tâche. Les mesures les plus utilisées dans la littérature d'AO sont les mesures classiques couramment appliquées en recherche d'information.

Nous utilisons les mesures habituelles de précision (P), rappel (R) et F1-mesure (F1), calculées au moyen des macro-mesures. Soit un problème de classification en n classes (c_i, $1 \leq i \leq n$), Les mesures P, R et F1 sont calculées comme suit :

— **Précision (P)** : représente la moyenne des précisions des n classes. Elle est calculée selon l'équation 4.1.

$$P = \frac{\sum_{i=1}^{n} P_i}{n} \tag{4.1}$$

avec :

$$P_i = \frac{\text{nombre de documents correctement attribués à la classe } c_i}{\text{nombre de documents attribués à la classe } c_i} \tag{4.2}$$

— **Rappel (R)** : représente la moyenne des rappels des n classes. Elle est calculée selon l'équation 4.3.

$$R = \frac{\sum_{i=1}^{n} R_i}{n} \tag{4.3}$$

avec :

$$R_i = \frac{\text{nombre de documents correctement attribués à la classe } c_i}{\text{nombre de documents appartenant à la classe } c_i} \tag{4.4}$$

— **F1 mesure (F1)** : représente la moyenne harmonique de la précision et du rappel. Elle mesure la performance du système et elle est calculée selon l'équation 4.5

$$F1 = \frac{2 \times P \times R}{P + R} \tag{4.5}$$

Enfin, nous utilisons la mesure d'**exactitude** (*Accuracy* **A**) pour évaluer nos systèmes de façon globale. Elle se calcule selon l'équation 4.6.

$$A = \frac{\text{nombre de documents correctement classés}}{\text{nombre total de documents}} \tag{4.6}$$

4.3 Premières expériences

Au début de la thèse, Le système de [Dahou *et al.*, 2016] (un CNN entrainé avec les embeddings *Dahou* (voir section 4.1.1 *CNN+Dahou*) était le système le plus performant sur le corpus LABR. Ainsi, nous considérons le système *CNN+Dahou* comme baseline. Ce système prend comme entrée les embeddings *Dahou*.

Nous supposons que la qualité des embeddings de mots utilisés pour entrainer le CNN affecte les résultats. Nous avons alors entrainé le CNN avec les différents embeddings de mots pré-entrainés existants (construits avec word2vec et fastText).

Nous présentons, dans la suite, les performances du CNN avec les différents ensembles d'embeddings. Nous proposons un protocole spécifique à la tâche d'analyse d'opinions pour la représentation vectorielle de documents.

4.3.1 Impact des pré-traitements du corpus

Nous avons classé les différentes opérations de pré-traitement par catégorie : réseaux sociaux, nombre, caractéristique de l'arabe, et normalisation. Le tableau 4.9 résume cette catégorisation.

Catégorie	Opérations
Réseaux sociaux	- supprimer les urls, les mentions et les hashtags - isoler les émoticônes - ne garder que 2 occurrences de caractères identiques consécutifs
Caractéristiques de l'arabe	- supprimer les mots non arabes, les voyelles et le caractère de prolongation
Normalisation	- supprimer les signes diacritiques pour la lettre *alif* - normaliser les signes de ponctuations
Nombre	- supprimer les nombres

TABLE 4.9 – Opérations de nettoyage du corpus par catégorie.

Afin de valider ce nettoyage, nous avons mesuré et comparé les performances de notre baseline (CNN+Dahou). Le tableau 4.10 rapporte les performances du CNN sur le corpus LABR dans le cadre de la classification binaire. La faible exactitude est obtenue sur le corpus LABR brut. L'application des opérations de nettoyage par catégorie améliore l'exactitude. La meilleure performance est obtenue sur le corpus nettoyé. Ceci valide nos opérations de nettoyage. Dans le reste de cette thèse, l'apprentissage et la prédiction avec nos systèmes neuronaux sont réalisés sur le corpus LABR nettoyé.

Catégorie	Dev	Test
Brut	88.8	88.9
Réseaus sociaux	88.9 ↗	89.1↗
Réseaus sociaux + caractéristique de l'arabe	88.9↘	89.2 ↗
Réseaus sociaux + caractéristique de l'arabe + Normalisation	89.7 ↗	89.6 ↗
Réseaus sociaux + caractéristique de l'arabe + Normalisation + Nombre	**89.9** ↗	**89.7** ↗
Réseaus sociaux + Normalisation + Nombre	89.6	89.1
Baseline	89.1	89.6

TABLE 4.10 – Exactitude (en %) du CNN sur LABR nettoyé par catégorie.

4.3.2 Premières évaluations du CNN

4.3.2.1 Impact du choix d'embeddings d'entrée sur le CNN

L'architecture CNN a été entrainée sur le corpus LABR avec les différents embeddings de mots pré-entrainés existants. Les tableaux 4.11 et 4.12 rapportent les performances du CNN dans les 3 cadres de classification (binaire, ternaire et quinaire) sur les corpus Dev et Test. Afin de facilité la lecture des très nombreux résultats, les meilleures performances sont mises en gras. Nous nous sommes basés sur les performances du CNN sur le corpus *Dev* de LABR pour choisir les meilleurs systèmes. En nous basant sur les performances du CNN sur le corpus *Dev*, nous déterminons les meilleurs systèmes dans les 3 cadres de classification. Nous considérons, tout d'abord, l'exactitude puis la mesure F1. Les systèmes choisis sont grisés.

Emb		2 classes				3 classes				5 classes			
		A	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**
word2vec	Dahou	89.1	82.67	75.87	79.12	72.7	60.00	52.37	55.93	55.1	50.68	46.91	48.72
	twt-cbow	89.1	82.70	75.55	78.97	72.5	59.51	52.24	55.64	55.5	50.71	**47.30**	48.95
	twt-sg	89.2	82.78	75.95	79.22	72.6	59.78	52.32	55.80	55.4	50.67	46.93	48.73
	wiki-cbow	89.3	83.00	**76.13**	79.42	72.5	59.49	52.20	55.60	55.1	50.19	46.59	48.33
	wiki-sg	89.1	82.61	75.98	79.16	72.5	59.66	52.11	55.63	55.1	50.36	46.75	48.49
	www-cbow	89.1	82.85	75.58	79.05	72.4	59.28	51.96	55.38	55.4	51.20	47.17	49.10
	www-sg	89.3	82.93	76.11	79.38	72.6	59.53	52.23	55.64	55.3	50.42	46.96	48.62
FastText	cc.ar	89.8	85.74	74.99	80.01	**73.5**	61.43	**53.80**	**57.36**	55.3	52.68	46.62	**49.46**
	cc.arz	**90.2**	**86.56**	75.86	**80.86**	72.3	**62.77**	48.95	55.01	**55.7**	**53.11**	46.00	49.30
	wiki.ar	89.1	82.66	75.48	78.91	72.5	59.68	52.32	55.76	55.1	50.18	46.60	48.32
	wiki.arz	89.5	83.66	76.39	79.86	72.6	59.27	52.08	55.44	**55.7**	50.87	**47.30**	49.02

TABLE 4.11 – Performances du CNN sur le corpus de validation Dev avec les différents embeddings.

Emb		2 classes				3 classes				5 classes			
		A	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**
word2vec	Dahou	89.6	82.78	76.76	79.66	73.2	60.11	52.86	56.25	49.9	44.77	43.29	44.02
	twt-cbow	89.6	82.76	76.73	79.63	73.3	60.37	52.90	56.39	50.1	45.51	43.74	44.61
	twt-sg	89.7	82.79	**76.90**	79.74	73.4	60.58	52.92	56.49	49.9	45.23	43.55	44.37
	wiki-cbow	89.6	82.56	76.64	79.49	73.2	60.02	52.68	56.11	49.9	44.98	43.58	44.27
	wiki-sg	89.6	82.54	76.74	79.53	73.2	60.20	52.68	56.19	49.6	44.68	43.22	43.94
	www-cbow	89.6	82.62	76.65	79.52	73.4	60.56	52.89	56.47	50.1	45.88	**43.78**	44.81
	www-sg	89.6	82.72	76.75	79.62	73.3	60.13	52.75	56.20	49.8	44.83	43.41	44.10
FastText	cc.ar	90.0	85.13	75.18	79.85	**74.1**	61.50	**54.72**	**57.91**	**50.2**	46.98	42.09	44.40
	cc.arz	**90.3**	**85.40**	76.16	**80.51**	73.1	**62.29**	49.16	54.95	**50.2**	**47.43**	42.76	**44.97**
	wiki.ar	89.6	82.84	76.59	79.59	73.2	60.05	52.79	56.19	50.0	45.56	43.66	44.59
	wiki.arz	89.6	82.70	76.85	79.67	73.4	60.56	53.05	56.55	49.8	45.28	**43.78**	44.52

TABLE 4.12 – Performances du CNN sur le corpus de test avec les différents embeddings.

Quelque soit le cadre de classification, les embeddings de type FastText donnent des performances du CNN plus élevées que celles obtenues avec les embeddings de type word2vec. Ces résultats se voient sur le corpus de *Dev* mais aussi sur le corpus de test. Ceci illustre la stabilité des meilleures

performances avec les systèmes choisis sur le *Dev*.
La tâche se complexifie avec le nombre de classes : plus le nombre de classes augmente, plus les performances décroîent.

		Classes prédites	
		négative	positive
Classes de référence	négative	932	757
	positive	246	8 355

TABLE 4.13 – Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.arz* dans le cadre de la classification binaire.

		Classes prédites		
		négative	neutre	positive
Classes de référence	négative	946	144	599
	neutre	372	330	1 658
	positive	267	234	8 100

TABLE 4.14 – Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.ar* dans le cadre de la classification ternaire.

		Classes prédites				
		très négative	négative	neutre	positive	très positive
Classes de référence	très négative	238	126	111	70	56
	négative	116	286	365	214	107
	neutre	48	156	791	905	460
	positive	18	53	568	1 734	1 465
	très positive	37	40	256	1 133	3 297

TABLE 4.15 – Matrice de confusion du CNN sur le corpus de test avec les embeddings *cc.arz* dans le cadre de la classification quinaire.

Les tableaux 4.13, 4.14 et 4.15 présentent les matrices de confusion des meilleurs systèmes par cadre de classification. Dans le cadre de la classification binaire, nous remarquons que le meilleur système a du mal à prédire la classe *négative* avec 757 documents prédit positifs alors qu'ils sont négatifs, soit 44.81% de la classe négative. Dans le cadre de la classification ternaire, nous constatons, en plus de la difficulté de prédiction de la classe négative avec 35.46% de négatifs prédit en positifs (soit 599 critiques), une difficulté de prédiction de la classe neutre avec seulement 13.98% prédit correctement et 70.25% de neutres prédit en positifs.
Enfin, les difficultés s'accroissent avec le nombre de classes. Nous remarquons notamment dans le cadre quinaire que de nombreux documents très positifs sont mal classés en positifs et inversement, c'est-à-dire que plusieurs documents positifs sont mal classés en très positifs. Ceci signifie qu'il y a une forte confusion entre les classes *très positive* et *positive*. La classe neutre s'étale largement sur les classes *positive* et *négative* et aussi un peu sur les classes *très positive* et *très négative*.

4.3.2.2 Premières analyses

Quelque soit le cadre de classification (binaire, ternaire ou quinaire), les systèmes présentent des difficultés de prédiction de certaines classes. Nous expliquons ces difficultés par deux raisons : la répartition des classes dans le corpus d'apprentissage et la nature de la classe.

Pour la première raison, les classes avec des difficultés de prédiction ont des proportions faibles dans le corpus d'apprentissage par rapport aux classes bien prédites. Par exemple, dans le cadre de la classification binaire, le nombre d'exemples négatifs représentent uniquement 16% du corpus d'apprentissage. Ainsi, le corpus d'apprentissage déséquilibré impacte les performances.
Il existe différentes techniques permettant d'équilibrer les corpus. Nous citons par exemple le sous-échantillonnage (*undersampling*), le sur-échantillonnage (*oversampling*) [Branco *et al.*, 2015], SMOTE (*Synthetic Minority Oversampling Technique*) [Chawla *et al.*, 2002], ROSE(*Random Oversampling Examples*) [He & Ma, 2013], *etc*. Il est crucial de tester les techniques d'équilibrage de corpus et mesurer son impact sur les performances. C'est une piste intéressante d'amélioration. Nous la gardons comme une perspective de cette thèse.

En ce qui concerne la deuxième raison, certaines classes peuvent être ambigües. En effet, nous rappelons que la classe est déterminée non par un annotateur linguistique professionel mais par la personne qui a laissé le commentaire. À ce moment là, elle avait à cœur de laisser son opinion sans penser à ce que le texte laissé soit cohérent avec le nombre d'étoiles choisi. Ainsi, il peut y avoir des divergences entre note (nombre d'étoiles) et contenu. De même, l'appréciation de *très positive* ou *positive* n'avait pas d'indication précise du site et dépend fortement du caractère de la personne.

Afin d'analyser les performances, nous donnons, dans la suite, quelques exemples de documents dans a) et, nous menons dans b) une analyse à base de lexiques polarisés.

a) Exemples de documents :
La neutralité d'une opinion n'est pas évidente, et elle demeure difficile à identifier mêmes par des êtres humains. Par exemple, les critiques 1, 2, 3 et 4 du tableau 4.16 sont très positifs alors qu'elles sont annotées neutres. De plus, la limite entre les classes *négative* et *très négative* (ou les classes *positive* et *très positive*) est floue. En effet, un commentaire peut contenir des parties négatives sur l'entité en question et pourtant son détenteur le note avec 5 étoiles (par exemple, le commentaire 5 dans le tableau 4.16). Pour finir, nous présentons l'exemple 6 où la personne, n'ayant pas lu le livre, a noté son commentaire avec *très positive*. Autrement dit, le contenu du document 6 ne reflète pas le nombre d'étoiles associé par son détenteur. Ces constatations nous ont conduit à analyser l'orientation en polarité des commentaires du corpus LABR.

b) Analyse basée sur un lexique polarisé :
Pour analyser plus finement la composition des commentaires, nous choisissons un lexique de même domaine que celui du corpus. En effet, nous nous appuyons sur les mots issus du lexique *LABR_lex* de [ElSahar & El-Beltagy, 2015] qui regroupe 873 expressions [4] dont la polarité est connue (398 positives et 475 négatives).

4. Une expression dans le lexique peut être constituée d'un ou plusieurs mots.

id	Document	Référence
1	تركيبها اللّغوي مبهر (sa structure linguistique est impressionnante)	* * * **(neutre)**
2	كتاب جميل يوضح امور لم اكن اعرفها عن القادة (Un beau livre qui explique des choses que je ne savais pas sur les leaders)	
3	ذكريات من الزمن الجميل (Souvenirs de la belle époque)	
4	مهم ان نقرا باستمرار ما يعيلنّا على اعادة تقيّم مستوانا الاخلاقي (Il est important de lire continuellement ce qui nous aide à réévaluer notre niveau moral)	
5	كتاب راءيع وسلس سان كان هناك ماخذ وهو انه غفل عن ذكر اناس كن لهم اثرهم الواضح في القرن السابق ... (Le livre est merveilleux et fluide, bien qu'il y ait un inconvénient, c'est qu'il a négligé de mentionner les personnes qui avaient une influence évidente au siècle précédent.) ...	* * * * * **(très positive)**
6	لسه مقرتوش (je ne l'ai pas encore lu)	

TABLE 4.16 – Exemples de documents dans le corpus LABR.

LABR_lex est alors constitué de 487 mots (235 positifs et 262 négatifs) pour un total de plus de 78k occurrences de mots dans le corpus LABR. Les mots de ce lexique constituent 2.4% des occurrences de mots contenus dans les documents positifs ou négatifs du corpus LABR (voir tableau 4.17). Ils sont autant présents dans les critiques positives que négatives. La majorité ($\geq$ 1.6%) de ces mots sont des mots positifs.

	LABR_lex$^+$	LABR_lex$^-$
Critiques positives	2.21	0.28
Critiques négatives	1.64	0.81

TABLE 4.17 – Répartition (en %) des mots polarisés de *LABR_lex* dans l'ensemble Train du LABR.

La difficulté de classification des critiques négatives peut donc être due à l'utilisation de figures de styles comme l'humour ou l'ironie qui implique qu'une expression positive est utilisée alors que le sens se veut négatif. Une autre explication à l'apparition de ces mots positifs dans une critique négative est qu'ils sont utilisés en conjonction avec un terme de négation. Nous avons par exemple remarqué que parmi les vingt mots les plus fréquents, trois étaient des termes de négation. Nous pensons également que la difficulté de classification des critiques négatives peut être fortement liée à la pertinence des *embeddings* d'entrée pour la tâche donnée. Nous proposons, dans la section 4.4, un protocole d'analyse afin d'étudier cette hypothèse.

4.3.3 Représentation des documents

La représentation des donnees d'apprentissage et son influence sur la qualité des systèmes reste toujours une question importante pour de nombreuses tâches en TALN. Il en est de même pour l'analyse d'opinions. Dans les expériences précédentes 4.3.2, la longueur de documents était fixée à 882 mots. Nous proposons dans cette section, un protocole spécifique de choix d'hyperparamètres ralatifs

à la représentation de documents impactant la tâche d'analyse d'opinions. Ce protocole regroupe le choix de la longueur de documents et le type de padding/truncating.

4.3.3.1 Longueur du document

Le réseau convolutif CNN prend comme entrée une matrice de taille fixe (voir section 3.2.1 du chapitre 3). Dans notre cas, la matrice représente le commentaire (document) : nombre de mots pris en compte $\times$ dimension des embeddings représentant les mots. Étant donné que la taille de l'embedding de mot est 300, il nous reste qu'à fixer le nombre de mots à prendre en compte pour représenter le document.
Il est d'usage dans la communauté scientifique d'utiliser la formule 4.7 pour fixer la longueur d'un document [5] [Grafarend, 2006]. Cette formule, dite règle empirique *empirical rule*, suppose que la distribution de longueur suit la loi gaussienne. En particulier, cette règle empirique prédit que 95% des observations se situent dans l'intervalle [moyenne - écart type , moyenne + écart type].

$$seuil = moyenne + 2 \times \text{écart type} \tag{4.7}$$

Nous avons utilisé le corpus d'apprentissage du LABR pour déterminer le nombre de mots moyen par documents (moyenne) et écart type (*écart type*). Ainsi, nous obtenons une moyenne égale à 64 mots et un écart type égal à 117.71. En appliquant la formule 4.7, nous obtenons donc un seuil de 300 mots. Nous déterminons donc que chaque document sera représenté par 300 mots et, ainsi chaque document sera représenté par une matrice de 300 mots par 300 composantes (taille de l'embedding de mot).

LABR	# document(longueur $\leqslant$ 300)	% document(longueur $\leqslant$ 300)
Train	27351	96.87%
Dev	4843	96.86%

TABLE 4.18 – Couverture de l'intégralité des documents de LABR avec le seuil 300

Le tableau 4.18 montre qu'au moins 96.8% des documents des corpus Train et Dev ont une taille qui varie entre 1 et 300. Nous déduisons que la très grande majorité des documents ont une taille inférieure ou égale à 300 mots et ainsi 4% des documents devront être tronqués. Nous définissons, dans la suite, comment représenter les documents en fonction de leur taille.

4.3.3.2 Type de padding/Truncating

Nous avons fixé, dans la section 4.3.3.1, la longueur de documents à 300 mots. Lorsque la taille du document est supérieure à celle fixée, il est nécessaire de couper (et ainsi ignorer) les mots supplémentaires : c'est le *truncating*. Lorsque les documents sont plus courts, il est nécessaire de combler la représentation du message par des zéros : c'est le *padding*. Or il existe trois façons de procéder à

5. https://en.wikipedia.org/wiki/68\T1\textendash95\T1\textendash99.7_rule

ce truncating/padding : soit couper/combler sur le début du message (pré), soit sur la fin du message (post), soit de manière égale sur les 2 extrémités.

Pour choisir le protocole padding/truncating le plus adéquat, nous avons procédé par une analyse des mots polarisés contenus dans les documents afin de déterminer le segment qui contient l'information la plus pertinente pour la classification. Dans le cadre de l'analyse d'opinions, cette information regroupe principalement les mots polarisés et les termes de négation qui sont souvent utilisés dans l'expression d'opinions. Pour déterminer la polarité d'un mot, le lexique *ArSentLex* a été utilisé (voir section 4.2.2). Pour les termes de négation, une liste prédéfinie regroupe 6 différents termes de négation : { ما, ليس, لن, لم, لا, غير}.

Nous avons effectué des statistiques sur les pourcentages de mots polarisés et de termes de négation pour mesurer l'informativité en polarité des différents segments du document. Nous avons divisé le document en trois segments et calculé le pourcentage de mots polarisés ou de terme de négation contenu dans chacun des trois segments. Ces statistiques sont reportées dans le tableau 4.19.

		1ier segment	**2ieme segment**	**3ieme segment**
Train	% mots positifs	16.33	0.73	0.82
	% mots négatifs	7.29	0.34	0.63
	% termes de négation	0.74	0.03	0.002
Dev	% mots positifs	16.27	0.72	0.82
	% mots négatifs	7.73	0.32	0.63
	% termes de négation	0.72	0.04	0.002

TABLE 4.19 – Informativité des différents segments du corpus Train et Dev de LABR

Une première remarque basée sur le tableau 4.19 porte sur l'informativité du premier tiers du document qui comprend le plus grand pourcentage de mots polarisés. Les deux autres tiers ne sont pas aussi informatifs que le premier. Nous pourrions donc déduire que les internautes expriment explicitement leurs opinions au début du commentaire. Ils se justifient par la suite de manière plus factuelle. Le premier tiers de chaque document semble donc contenir de l'information pertinente pour la classification en polarité. Le post-padding/post-truncating semble ainsi être adapté à l'analyse d'opinion du corpus LABR. Si le document comprend plus de 300 mots, la fin de celui-ci sera donc coupé. S'il est plus petit, il sera complété par la composante 0 autant que nécessaire.

4.3.3.3 Impact sur les performances

Nous présentons dans cette section les performances des meilleurs systèmes obtenus avec les embeddings de type word2vec et FastText présentés dans la section 4.3.2 avec adaptation de la longueur de documents à 300 et un post-padding. Notons par :

— *CNN_DAHOU* : l'architecture CNN avec la longueur de document égale à 882 et le pré-padding comme proposée par Dahou *et al.* [2016] et dont les performances sont présentées dans la section 4.3.2.1

— *CNN_300-post* : l'architecture du CNN avec application du protocole de représentation de documents (présenté dans les sections 4.3.3.1 et 4.3.3.2), c'est-à-dire avec la longueur de document égale à 300 et le post-padding.

Le tableau 4.20 rapporte les performances du CNN sur le corpus de test. Les lignes grisées représentent les systèmes choisis en se basant sur les meilleures performances obtenues sur le corpus de Dev. Le tableau A.1 de l'annexe A représente les performances sur le corpus de *Dev*.

Nous remarquons, tout d'abord, une stabilité des meilleures performances sur les corpus de *Dev* et de test dans les cadres de classification binaire et ternaire. Ce n'est pas le cas pour le cadre quinaire : le meilleur système sur le *Dev* est obtenu avec les embeddings *twt-cbow*, alors que le meilleur système sur le corpus de test est obtenu avec les embeddings *cc.arz*.

	Emb	***CNN_300-post***				***CNN_DAHOU***		**Impact**
		A	**P**	**R**	**F1**	**A**	**inter_conf**	
2 classes	wiki-cbow	89.6	83.02	76.12	79.42	89.6	[89.0, 90.1]	0
	cc.arz	**90.3**	85.91	76.02	80.67	90.3	[89.7, 90.8]	0
3 classes	Dahou	73.6	60.90	52.91	56.62	73.2	[72.4, 73.9]	+0.4
	cc.ar	**74.4**	61.82	55.25	58.35	74.1	[73.3, 74.8]	+0.3
5 classes	twt-cbow	49.7	44.53	42.99	43.75	50.1	[49.2, 50.9]	-0.4
	cc.arz	**50.5**	46.75	42.23	44.38	50.2	[49.3, 51.0]	+0.3

TABLE 4.20 – Performances du CNN sur le corpus de Test avec longueur 300.

Les performances du *CNN_300-post* sont proches de celles de *CNN_DAHOU*. Au niveau exactitude, on note des gains et des pertes selon les embeddings pré-entrainés utilisés. Néanmoins, les exactitudes de *CNN_300-post* sont incluses dans les intervalles de confiance des performances de *CNN_DAHOU*. On pourrait ainsi juger les gains et les pertes comme négligeables et non significatifs. Une première raison évidente à cela revient au nombre de documents impactés par notre protocole de choix de représentation de documents. En effet, les documents tronqués dans le corpus de test sont au nombre 255 (2%) : 224 documents positifs et 31 négatifs. En contre partie et dans le cas où la longueur est égale 882, le nombre de documents tronqués dans le corpus de test est 632 (5%).
Finalement, il est toujours mieux de mettre en place un protocole de choix de paramètres que de les fixer au hasard. D'où l'intérêt de notre protocole de choix de représentation de documents.

4.4 Embeddings de mots : étude qualitative

Afin d'évaluer la pertinence dans le cadre spécifique de la tâche d'analyse d'opinions des représentations de mots dans un espace continu, nous proposons deux points de vue différents d'observations de l'espace des embeddings.

4.4.1 Couverture

Dans un premier temps, nous proposons de calculer la couverture des mots du corpus LABR par les projections existantes dans l'un des 5 espaces d'embeddings (de type word2vec et FastText) les

plus performants dans les trois cadre de classification, à savoir *Dahou*, *wiki-cbow*, *twt-cbow*, *cc.ar* et *cc.arz*. En effet, les mots du document sont représentés par des embeddings pré-existants, mais il peut arriver qu'un embedding d'un mot donné n'existe pas dans l'espace d'embeddings. À ce moment là, il est initialisé aléatoirement.

Pour ce faire, nous avons considéré d'une part tous les mots puis d'autre part les occurences de mots du corpus LABR puis d'autre part le vocabulaire du corpus LABR. Pour chacun, nous avons alors vérifié la présence du mot ou de l'occurence du mot dans l'espace d'embedding considéré. Nous avons ainsi pu calculer les couvertures du corpus LABR par les différents espaces d'embeddings. Les résultats sont présentés dans les tableaux 4.21 et 4.22.

Corpus LABR	**word2vec**			**FastText**	
	Dahou	wiki-cbow	twt-cbow	cc.ar	cc.arz
tous	67.33	61.53	60.72	100	100
occurrence $\geqslant 5$	71.07	68.06	66.04	100	100

TABLE 4.21 – Couverture (en %) du corpus LABR par les d'embeddings de type word2vec et FastText les plus performants.

Corpus LABR	**word2vec**			**FastText**	
	Dahou	wiki-cbow	twt-cbow	cc.ar	cc.arz
tous	40.97	19.16	22.11	100	100
occurrence $\geqslant 5$	64.89	53.48	57	100	100

TABLE 4.22 – Couverture (en %) du vocabulaire de LABR par les d'embeddings de type word2vec et FastText les plus performants.

Pour les espaces d'embeddings de type word2vec, nous remarquons que la couverture du corpus se situe aux alentours de 60% quels que soient l'espace considéré. La couverture augmente de 3 à 6 points si on ne considère que les mots fréquents dont le nombre d'occurrences est $\geqslant 5$. Au niveau du vocabulaire, plus de 53% des mots fréquents sont couverts alors que la couverture du vocabulaire chute à 20% si on considère tous les mots. Ceci indique que la grande majorité des mots du corpus LABR n'ayant pas d'*embeddings* dans les modèles disponibles regroupe des mots peu fréquents. Les embeddings *Dahou* ont la couverture la plus grande alors qu'ils ne donnent pas forcément la meilleure performance. Ainsi, bien que la couverture ne soit pas très grande elle semble suffisante pour la classification.

Les espaces d'embeddings de type FastText couvrent à 100% le corpus LABR. Ceci s'explique par mode de construction d'embeddings vec FastText. En effet, si le mot n'existe pas dans le vocabulaire et pour lui attribuer un embedding, FastText se base sur les sous-unités dont leur meilleure combinaison constitue le mot.

La couverture est un aspect important dans la classification. Plus la couverture est élevée, plus les performances sont meilleures. En effet, les meilleures performances dans les 3 cadres de classification

sont obtenues avec des embeddings de type FastText (*cc.ar* et *cc.arz*) couvrant à 100% le corpus LABR.

Par ailleurs, les embeddings FastText couvrent 100% le corpus LABR, alors que les embeddings word2vec couvrent au maximum 67% du corpus. Autrement dit, il existe environ 30 points de différence en couverture entre les embeddings FastText et word2vec. En contre partie et au niveau performance, les résultats du CNN avec les embeddings FastText apportent en moyenne un gain de 1 point par rapport à celles obtenues avec les embeddings word2vec. Le gain en performance est donc relativement petit par rapport à la différence en couverture. Dans ce contexte, un point très important à soulever est la version de l'architecture CNN. En effet, toutes les expériences réalisées jusqu'à maintenant sont basées sur la version non statique du CNN. C'est-à-dire que les embeddings d'entrée au CNN sont mis à jour lors de l'apprentissage. Ceci signifie que les embeddings de mots non couverts, initialisés au début de façon aléatoire, sont modifiés lors de l'entraînement du CNN en prenant en compte la tâche de classification en polarité. Nous étudions de près les versions statique et non statique du CNN et leurs impacts sur les performances de classification dans le chapitre 5.

4.4.2 Étude du voisinage

Dans un second temps, nous proposons d'étudier la polarité des mots voisins d'un mot polarisé, en considérant leurs *embeddings* dans chacun des espaces considérés, pout les mots polarisés. Pour chaque mot palarisé du lexique, son ensemble des n plus proches mots polarisés voisins (Top_n) dans l'espace d'embeddings, est considéré selon la similarité cosinus. C'est-à-dire que dans l'espace d'embeddings considéré, seuls les mots du lexique polarisé sont conservés. Puis, pour chaque mot positif (respectivement négatif), nous allons étudier les n plus proches voisins et considéré uniquement les mots positifs (respectivement négatifs) pour calculer le ratio de positivité (respectivement négativité) selon l'équation 5.1. Les plus proches voisins sont considérés selon la similarité cosinus entre les embeddings des 2 mots dans l'espace considéré.

$$\%^{+}_{Top_n} = 100 \times \frac{\sum_{mot_i \in \{lexique^+\}} \#mot^{lexique^+}_{i,Top_n}}{n \times \#lexique^+} \tag{4.8}$$

avec : n le nombre de mots voisins considérés ; $\#mot^{lexique^+}_{i,Top_n}$ le nombre de mots positifs parmi les n plus proches voisins du mot i du corpus $lexique^+$; $\#lexique^+$ le nombre de mots positifs dans $lexique$.

Nous calculons également un ratio de *négativité* selon la même formule en ne considérant que les mots négatifs. Nous considérons qu'une représentation pertinente des mots dans un espace continu pour la tâche de détection d'opinions projetterait les mots positifs dans la même zone et les mots négatifs dans une autre zone. On observerait alors un ratio proche de 100%.

La Table 4.23 montre les calculs des ratios de positivité et de négativité effectué sur le lexique *LABR_lex* avec les embeddings de type word2vec et FastText. Pour les embeddings de type word2vec,

		word2vec			**FastText**	
		Dahou	wiki-cbow	twt-cbow	cc.ar	cc.arz
$\%^{+}_{Top_n}$	n=2	43.13	38.83	41.79	100	100
	n=5	68.10 ↗	58.75 ↗	63.28 ↗	40.51 ↘	40.51 ↘
	n=10	73.46 ↗	63.39 ↗	68.43 ↗	30.51 ↘	30.51 ↘
#LABR_lex^{+}		153	112	134	235	235
$\%^{-}_{Top_n}$	n=2	34.88	31.77	38.75	0	0
	n=5	13.95 ↘	12.71 ↘	15.5 ↘	20 ↗	20 ↗
	n=10	6.97 ↘	6.35 ↘	7.75 ↘	40 ↗	40 ↗
#LABR_lex^{-}		172	107	160	262	262

TABLE 4.23 – Ratios de *positivité* (respectivement *négativité*) des mots positifs (respectivement négatifs) dont l'embedding existe à la fois dans *LABR_lex* et le corpus d'*embeddings* considéré.

nous constatons que plus le voisinage considéré est large, plus le ratio de positivité est grand. Ceci signifie que les mots positifs sont de plus en plus entourés par des mots positifs du lexique. En revanche, pour le ratio de négativité, nous constatons que plus le voisinage est large, moins le mot négatif est entouré de mots négatifs.
Etant donné que seuls les mots polarisés sont considérés, ceci signifie que les mots négatifs sont de plus en plus entourés par des mots positifs du lexique. Ces observations se vérifient pour les différents espaces d'embeddings. La polarité négative semble donc diffusée dans l'espace de représentations utilisé. Ceci appuie notre hypothèse d'un espace continu non adapté au cadre de la détection d'opinions, notamment pour représenter les mots négatifs. Cette observation peut également expliquer la difficulté de classification des commentaires négatifs.

Pour les embeddings de type FastText, nous constatons que plus le voisinage est large, plus le ratio de positivité est petit. Ceci signifie que dans un voisinage restreint, les mots positifs sont entourés par des mots positifs formant ainsi des petites zones composées de mots positifs. En revanche, pour le ratio de négativité, nous constatons que plus le voisinage est large, plus le ratio de négativité est grand. Ceci signifie que les mots négatifs sont dispersés dans l'espace.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une étude préliminaire d'un système neuronal pour l'analyse d'opinions en arabe. Nous avons testé un réseau convolutif CNN dans 3 cadres de classification : binaire, ternaire et quinaire. Nous avons réglé dans un premier temps l'architecture CNN selon [Dahou *et al.*, 2016] et avons testé différents ensembles d'embeddings de mots pré-entrainés.
Puis, nous avons proposé un protocole spécifique à l'analyse d'opinions permettant de modifier quelques hyper-paramètres du CNN afin de choisir la longueur du document et le padding. Les performances du CNN varient peu en fonction de la nature des ensembles d'embeddings de mots (word2vec ou fastText). Nous avons néamoins choisi 300 comme longueur du document et le post

padding/truncating qui présentent des résultats similaires mais avec un gain en temps.
Enfin, nous avons mené une étude qualitative des espaces d'embeddings de mots pré-entrainés utilisés. Nous avons émis l'hypothèse que la qualité de l'espace d'embeddings vis à vis de la tâche d'opinions pouvait impacter les performances, notamment pour les documents négatifs. La couverture du corpus par les espaces d'embeddings impacte également les résultats avec de meilleurs résultats lorsque la couverture est meilleure.

Nous avons utilisé dans ce chapitre des embeddings de mots. Or, un mot arabe est complexe compte tenu de l'agglutination de cette langue. Nous allons voir dans le chapitre suivant comment prendre en compte la spécificité de la langue arabe dans l'espace d'embeddings et construire des embeddings spécifiques à la langue arabe, au lieu d'utiliser des embeddings génériques pré-entrainés existants.

Chapitre 5

Embeddings spécifiques à la langue arabe

Sommaire

5.1 Motivations 104
5.2 Analyse extrinsèque des embeddings spécifiques à l'arabe 105
5.2.1 Évaluation des embeddings simples 105
5.2.2 Évaluation de combinaison des sorties de systèmes 118
5.2.3 Évaluation de combinaison des entrées de systèmes 121
5.3 Analyse qualitative des embeddings spécifiques à l'arabe 123
5.3.1 Méthodologie 124
5.3.2 Résultats 126
5.3.3 Discussion 129
5.4 Conclusion 131

La majorité des travaux récents sur l'analyse d'opinions en arabe (AOA) nécessite des embeddings de mots comme entrée. Ces embeddings considèrent les mots comme étant des *unités séparées par des espaces* afin de capturer, dans l'espace de projection, des similarités sémantiques et syntaxiques. La qualité d'un espace d'embeddings nécessite de grands corpus de sorte que chaque mot du vocabulaire apparaisse plusieurs fois dans des contextes différents. La langue arabe se caractérise par son agglutination et sa richesse morphologique, conduisant à un vocabulaire très épars. Pour cela, la prise en compte de la complexité des mots arabes semble être essentiel dans le processus de construction des espaces d'embeddings. Actuellement, les espaces d'embeddings pré-entrainés existants représentent un mot arabe sans considération des caractéristiques d'agglutination et de la richesse morphologique de l'arabe. Nous proposons de construire des espaces d'embeddings prenant en compte la spécificité de la langue arabe et de les évaluer pour la tâche d'analyse d'opinions [1].

1. Les embeddings proposés peuvent être utilisés dans d'autres tâches NLP tel que le résumé automatique, *etc.*

Ce chapitre est organisé comme suit. Nous décrivons d'abord dans la section 5.1 les motivations de construction d'embeddings spécifiques à l'arabe. Nous présentons et évaluons par la suite tels embeddings dans la section 5.2. Puis, nous menons une analyse qualitative de ces embeddings pour la tâche d'analyse d'opinions en arabe dans la section 5.3.

5.1 Motivations

Les travaux à base de réseaux de neurones en AOA utilisent des embeddings de mots pré-entrainés. Nous avons testé, dans le chapitre 4, les différents espaces d'embeddings de mots pré-entrainés disponibles en langue arabe de type word2vec et FastText. Néanmoins, des améliorations peuvent encore être obtenues si on prend en compte les spécificités d'agglutination et de richesse morphologique de la langue arabe. Notamment, si on considère que la définition d'un mot, au sens graphique, est une séquence de caractères délimitée par deux séparateurs (blanc ou autre marqueur de séparation, tel que la ponctuation), alors un mot en arabe peut avoir une structure très complexe. En effet, ce mot peut être segmenté en proclitique(s), forme fléchie et enclitique(s).
Par exemple, le mot اسيعجبه /AsyEjbh/ (est-ce qu'il va lui plaire), se compose, selon le processus de segmentation (présenté dans la section 2.2.1 du chapitre 2), d'une particule d'interrogation ا et de futur س, de la forme fléchie يعجب et du pronom relatif ه qui sont tous agglutinés. Le tableau 5.1 montre que les trois mots اسيعجبه, فسيعجبك et و يعجبهم partagent la même forme fléchie يعجب.
En plus du processus de segmentation, il existe d'autres processus traitant l'agglutination et la richesse morphologique de la langue arabe, à savoir la lemmatisation, le stemming et le light stemming [2]. Pour chaque processus correspond une unité lexicale. Le tableau 5.2 montre, par exemple, que les mots partageant le même lemme جميل (beau) ne changent pas de polarité (positive).
Dans cette perspective, nous supposons qu'une décomposition en éléments simples du mot complexe pourrait réduire la taille du vocabulaire arabe (comprenant plusieurs centaines de millions de mots) et augmenter les occurrences de chacun des éléments et leurs contextes. Ceci pourrait améliorer la qualité des embeddings.

Mot	**Traduction**	**Décomposition**		
		Proclitiques	**Forme fléchie**	**Enclitiques**
اسيعجبه	est-ce qu'il va lui plaire ?	س + ا	يعجب	ه
فسيعجبك	il va te plaire	س + ف		ك
و يعجبهم	et il va leur plaire	و		هم

TABLE 5.1 – Exemples de mots partageant la même forme fléchie.

L'objectif de ce chapitre consiste donc à vérifier et valider cette hypothèse. L'idée consiste à explorer les embeddings spécifiques à la langue arabe afin de trouver la meilleure représentation de

2. Ces processus de segmentation, lemmatisation, stemming et light stemming ont été définis en section 2.2 dans le chapitre 2.

Mot			Lemme
	Genre	Nombre	
جميلان	male	duel	جميل (beau)
جميلون	male	pluriel	
جميلات	femelle	pluriel	
احبّ	male and femelle	singulier	احبّ (aimer)
يحبّون	male	pluriel	
تحبّان	male and femelle	duel	

TABLE 5.2 – Exemples montrant la réduction en vocabulaire entre mots et lemmes.

l'unité lexicale pour l'analyse d'opinions.
Nous détaillons, dans la section 5.2, la méthodologie de construction de tels embeddings, baptisé "embeddings spécifiques à l'arabe" (*ArEmb*).

5.2 Analyse extrinsèque des embeddings spécifiques à l'arabe

Nous présentons, dans un premier temps, le protocole de construction des embeddings spécifiques à l'arabe et leur évaluation dans la section 5.2.1. Or, il est d'usage dans la communauté scientifique d'utiliser des méthodes de combinaison afin d'améliorer les performances [El-Halees, 2011; Ghannay *et al.*, 2016]. Nous appliquons, dans un deuxième temps, deux méthodes de combinaison. La première, présentée dans la section 5.2.2, consiste à combiner les sorties de différents systèmes. Quant à la deuxième méthode (présentée dans la section 5.2.3), elle consiste à combiner les entrées de systèmes.

5.2.1 Évaluation des embeddings simples

Les processus de prétraitement de la langue arabe sont : segmentation, lemmatisation, light stemming et stemming. Ces derniers permettent d'obtenir, pour un mot donné, des représentations différentes. Nous considérons ainsi les représentations de mots suivantes :

— mot : unité séparé par des espaces
— token : forme fléchie ou clitiques (les unités obtenues après segmentation)
— token\clitiques : forme fléchie
— lemme : forme canonique
— stemme : racine du mot
— light stemme : stemme + infixes

L'idée consiste à explorer les embeddings spécifiques à la langue arabe afin de trouver la meilleure représentation pour l'analyse d'opinions. À notre connaissance, [Salama *et al.*, 2018] est le seul travail traitant de la spécificité de l'arabe dans les espaces d'embeddings. Ce travail a étudié l'effet de

l'incorporation d'informations morphologiques aux embeddings de mots de 2 manières : (i) inclure des balises POS avec des mots avant la construction d'embeddings et, (ii) effectuer une abstraction (basée sur les lemmes) des plongements morphologiques obtenus en (i). Dans cette thèse, nous avons procédé différemment et nous avons construit des embeddings pour différentes unités lexicales arabes.

Dans cette section, nous présentons notre protocle de mise en place des embeddings spécifiques à l'arabe. Nous commençons tout d'abord par présenter le protocole de construction des différents espaces d'embeddings *ArEmb*, grâce aux deux techniques de construction word2vec et FastText dans la section 5.2.1.1. Puis, nous présentons leurs résultats d'évaluation dans la section 5.2.1.2.

5.2.1.1 Protocole

Dans notre protocole, nous commençons par présenter, dans la section 5.2.1.1.a, notre méthodologie de construction des embeddings spécifiques à l'arabe. Puis, nous présentons les deux architectures neuronales que nous avons utilisées, à base l'une d'un CNN et l'autre d'un BiLSTM. Ces deux dernières sont présentées dans la section 5.2.1.1.b, et elles sont utilisées dans le cadre de la classification binaire. Nous choisissons deux réseaux de nature différente et nous espérons déduire l'unité lexicale adéquate à l'AOA. Dans l'idéal, nous souhaitons obtenir les meilleures performances avec la même unité lexicale pour les deux réseaux neuronaux.

Nous utilisons le corpus LABR, présenté dans la section 4.2.1 du chapitre 4 comme corpus d'apprentissage. Nous nous limitons à la classification binaire. Il nous importe à indiquer que le corpus LABR contient de l'ASM et l'AD. Or les outils TAL, dédiés à la segmentation, la lemmatisation, le stemming et le light stemming s'appliquent à l'ASM. Nous devons dans ce cas calculer le pourcentage de l'ASM dans le corpus LABR. Si ce dernier est majoritairement en ASM, alors on peut utiliser nos outils TAL tout en étant conscient des erreurs dûes à l'application de ces outils sur les mots en AD. Si l'AD est très présent dans le corpus, alors il vaut mieux appliquer d'autres outils dédiés à l'AD, or la recherche dans ce sens est encore ses débuts et tels outils, s'ils existent, ne sont pas performants. Notre méthode de calcul de l'ASM dans le corpus LABR est détaillée dans la section 5.2.1.1.c.

5.2.1.1.a Construction des embeddings spécifiques à l'arabe

Nous présentons, dans la suite, la démarche de construction des embeddings spécifiques à l'arabe. Nous proposons d'utiliser comme corpus d'entraînement des embeddings spécifiques une concaténation de différents corpus arabes existants portant sur la tâche de détection d'opinions ou sur des articles de presse. Nous disposons des quatre corpus suivants :

— *BRAD* [Elnagar *et al.*, 2018b] regroupant 510k critiques sur livres
— *HARD* [Elnagar *et al.*, 2018a] constitué de 373K commentaires sur hôtels
— *LABR* [Nabil *et al.*, 2014] en considérant uniquement l'ensemble d'apprentissage formé de 23K de critiques sur livres
— *AbuElKhair* [El-Khair, 2016] regroupant 5 222k articles de presse

Le corpus résultant obtenu après concaténation est noté *Global*. Les corpus *BRAD*, *HARD* et *LABR* regroupent des commentaires sur des entités (livres ou hôtels). Ces trois corpus sont dédiés à la tâche d'analyse d'opinions. Quant au corpus *AbuElKhair*, contenant des articles de presse, il est générique et il n'est pas conçu pour l'AO. Nous sommes conscients que les articles de presse peuvent contenir des opinions. Nous nous sommes basés, pour la catégorisation en corpus spécifique à l'AO et corpus générique, sur la tâche pour laquelle le corpus a été conçu. Les corpus spécifiques à l'AO permettent d'avoir une idée sur les mots généralement utilisés en AO et leurs contextes. Quant au corpus générique, il permet d'augmenter éventuellement la taille du vocabulaire considéré et les contextes des mots qui le composent.

Nous avons commencé tout d'abord par nettoyer le corpus *Global*. Nous avons appliqué le même nettoyage que celui appliqué sur le corpus LABR présenté dans la section 4.2.1.3 du chapitre 4. Nous avons supprimé les urls, les mentions, les hashtags, les nombres, les signes de ponctuation et les mots non arabes. Nous avons normalisé les caractères ٱ, آ à un simple alif ا. Nous avons également supprimé les voyelles courtes et les signes diacritiques (*soukoun* et *chadda*). Ce nettoyage permet de normaliser les différentes possibilités d'écriture de mots en arabe ce qui permet de réduire la taille du vocabulaire et ainsi diminuer le nombre d'hapax. Le tableau 5.3 montre que le vocabulaire du corpus *Global* est beaucoup plus petit après nettoyage. En effet, le vocabulaire du corpus contient 12 359 750 mots avant nettoyage et 4 135 532 mots après nettoyage. Nous observons également que le nombre d'occurences après nettoyage (1 358 346 146) est plus grand que celui avant nettoyage (1 236 999 410). Ceci signifie une augmentation du nombre d'occurences des mots après nettoyage et par conséquent une augmentation du nombre de contextes des mots permettant ainsi d'obtenir des embeddings de meilleure qualité.

Taille	**Avant nettoyage**	**Après nettoyage**
Vocabulaire	12 359 750	4 135 532
Corpus	1 236 999 410	1 358 346 146

TABLE 5.3 – Statistiques sur le corpus *Global* avant et après nettoyage.

Le corpus *Global* représente le corpus au niveau mot. Pour les autre niveaux de représentation, nous avons appliqué, sur le corpus *Global* nettoyé les outils TAL de segmentation, lemmatisation, light stemming et stemming pour obtenir respectivement 4 corpus : un par unité lexicale, avec unité lexicale $\in$ {token, lemme, light stem, stem}. Nous avons appliqué à notre corpus *Global* :

— le tokeniseur de Farasa [3] pour obtenir les représentation du corpus aux niveaux token
— le lemmatiseur de Farasa [4] pour obtenir la représentation du corpus *Global* au niveau lemme
— le light stemmer de Motaz SAAD [5] pour obtenir la représentation du corpus *Global* au niveau light stemme

3. http ://qatsdemo.cloudapp.net/farasa/
4. http ://qatsdemo.cloudapp.net/farasa/
5. https ://github.com/motazsaad/arabic-light-stemming-py

Plusieurs outils TAL existent pour le traitement de la langue arabe. Nous choisissons les outils les plus connus, récemment développés et les plus performants. L'outil Farasa est développé en 2016 il est plus performant que les autres outils pour la segmentation et la lemmatisation [Abdelali *et al.*, 2016]. Le light stemmer est également connu par sa performance [6] [Saad, 2011].
La forme fléchie du mot représente l'unité lexicale de type *token\clitiques*. Pour obtenir l'unité *token\clitiques*, nous avons supprimé du corpus *Global* segmenté :

— les proclitiques : و, ف, ب, ل, س, أ, ك, ال
— les enclitiques : ي, ك, ه, ها, هما, هم, هن, كما, كم, كن, تما, تم, تن, نا, ة, ون, ان, ات, وا

Contrairement aux affixes, les clitiques [Alotaiby *et al.*, 2010] n'affectent pas la signification des mots et ils ne contribuent pas généralement à la détection de polarité. Par exemple, la suppression du déterminant ال du nom défini الجمال (la beauté) donne le nom indéfini جمال (une beauté) et elle n'influe pas sur le sens du nom qui se dit généralement sur ce qui "*touche et charme les sens, l'esprit, l'âme, de ce qui est excellent en son genre*" (wikipedia). De plus, le déterminant ال n'intervient pas dans la polarité du nom : les deux noms défini الجمال et indéfini جمال reflètent la polarité positive. Ainsi, la suppression des clitiques pourrait améliorer la qualité des embeddings pour l'AOA.

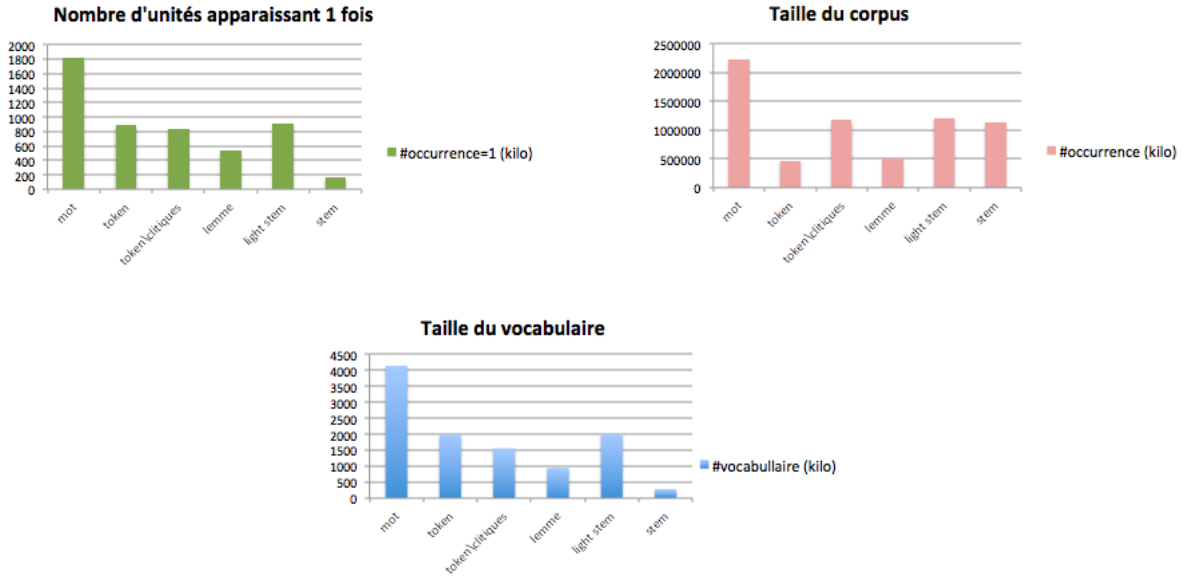


FIGURE 5.1 – Statistiques sur le corpus Global au niveau des différentes unités lexicales envisagées (mot, token, token\clitiques, lemme, light stemme et stemme) : taille du vocabulaire, taille du corpus et nombre d'unités apparaissant une fois.

Les histogrammes de la figure 5.1 montrent les statistiques sur le corpus *Global* nettoyé. Comme prévu, l'application d'outils TAL permet de réduire le vocabulaire. En effet, la taille du vocabulaire des unités lexicales autre que mot représente au maximum la moitié de la taille du vocabulaire des

6. https://weka.sourceforge.io/packageMetaData/ArabicStemmers_LightStemmers/index.html

mots. De plus, le nombre de lemmes apparaissant une seule fois est très faible par rapport au nombre de mots apparaissant une fois dans le corpus. Ainsi, l'application des outils TAL aide réellement à réduire le vocabulaire épars de l'arabe.

L'application des outils TAL sur *Global* permet d'obtenir 6 corpus : un corpus par unité lexicale. Pour chaque unité lexicale, nous avons construit deux ensembles d'embeddings : un obtenu avec word2vec et l'autre obtenu avec FastText. Nous avons choisi de garder la même dimension des embeddings pré-entrainés existants afin de comparer nos embeddings spécifiques à l'arabe avec les autres existants. Nous avons donc construit des embeddings de dimension 300.

Pour entraîner word2vec, nous avons choisi la version skip-gram (celle-ci étant plus performante que la version CBOW [Kim, 2014a; Dahou *et al.*, 2016; Elrazzaz *et al.*, 2017]). Nous avons gardé les valeurs de paramètres utilisés dans [Dahou *et al.*, 2016] et [Soliman *et al.*, 2017] : une fenêtre de taille 5 et un negative sampling de 10. Nous avons choisi de construire un embedding pour toute unité dans le corpus. Pour cela, nous avons fixé le paramètre min_freq à 1. Pour le paramètre relatif au nombre d'époques, nous avons choisi la valeur 30 et nous avons sauvegardé l'ensemble d'embeddings obtenu à chaque époque. Le nombre d'époques le plus utilisé dans la littérature est 5. Il a été fixé à 3 dans [Dahou *et al.*, 2016] et à 5 dans [Soliman *et al.*, 2017]. Les performances de systèmes reportées plus tard dans ce chapitre sont les meilleures obtenues avec les 30 ensembles d'embeddings. Nous allons étudier dans la section 5.3 l'impact du nombre d'époques sur les performances des systèmes.

Pour entraîner FastText, nous avons effectué des expériences pour choisir la valeur n du n-gramme contrôlée par les deux paramètres : *min_n* pour le nombre de n-grammes minimal et *max_n* pour le nombre de n-grammes maximal. Cela signifie que FastText va construire, en plus des embeddings d'unités complètes, des embeddings pour toutes les sous-unités de longueur n-grammes, avec $min\_n \leq n \leq max\_n$. La valeur par défaut de n est entre 3 et 6. En plus du couple (3,6), nous avons construit des modèles *.bin* obtenus avec les couples (min_n, max_n) suivants : (2,6), (4,6), (4,7), (4,7) et (3,7). Les meilleures performances de systèmes, reportées dans ce chapitre, sont obtenues avec le modèle entrainé avec le couple (3,6).

Nous avons construit 12 espaces d'embeddings (de dimension 300) dédiés aux 6 différentes unités lexicales (mot, token, token\clitiques, lemme, light stemmes et stemme) avec word2vec (version skip-gram) et FastText. La table 5.4 reporte la taille de chaque espace.

Unité	**mot**	**token**	**token\clitiques**	**lemme**	**light stemme**	**stemme**
Taille	4 135 532	1 980 255	1 555 872	946 171	1 997 601	280 508
Taux du compact (%)	100	47.88	37.62	22.87	48.30	6.78

TABLE 5.4 – Taille des espaces d'embeddings spécifiques proposés pour l'arabe.

La table 5.4 montre que le vocabulaire au niveau mot est très grand. Le corpus *Global* au niveau mot est le plus grand, il contient plus de mots que les autres corpus relatifs aux autres unités lexicales. Le plus petit corpus est celui au niveau stemme avec une taille de 6.78 fois plus petite.

Comme l'aspect épars du vocabulaire arabe a été réduit par application des outils TAL, nous prévoyons que les performances obtenues avec les mots seraient moins élevées que celles obtenues avec

les autres unités lexicales. Nous nous attendons également à des résultats moins bons avec les stemmes car le taux du compact est très petit, signifiant une perte importante de l'information sémantique.

5.2.1.1.b Architectures neuronales

Dans le but d'évaluer nos embeddings *ArEmb* et généraliser des conclusions, nous choisissons de tester deux architectures neuronales différentes. Ces architectures prennent en entrée les embeddings spécifiques à l'arabe. Pour chaque unité lexicale, chaque architecture utilise l'ensemble d'embeddings correspondant. Nous présentons, dans la suite, ces deux architectures neuronales.

La première architecture est basée sur le réseau convolutif CNN. Nous utilisons l'architecture CNN présentée dans le chapitre 4 avec application de notre protocole de représentation du document, c'est-à-dire, une longueur du document égale à 300 avec un post-padding et un post-truncating. Ces choix ont été justifiés dans le chapitre 4.

La deuxième architecture neuronale est basée sur le réseau récurrrent BiLSTM (voir section 3.3 du chapitre 3). Nous utilisons l'architecture neuronale BiLSTM. Nous choisissons de la renforcer avec un CNN pour obtenir notre architecture finale *BiLSTM*.

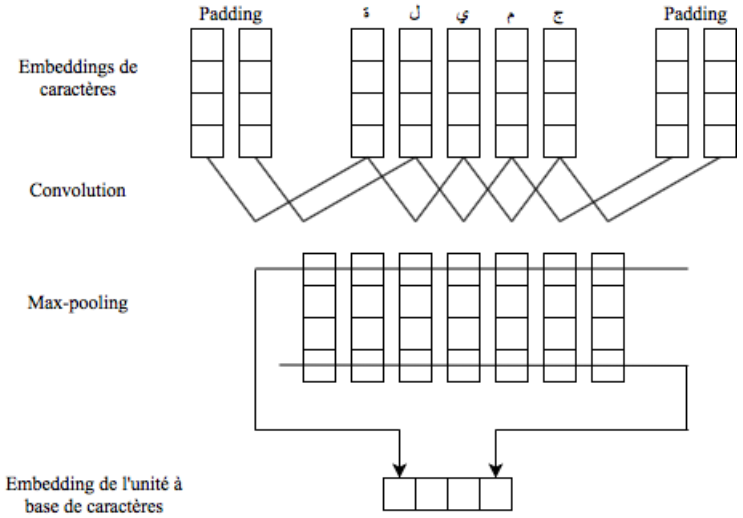


FIGURE 5.2 – Processus d'extraction d'embedding d'unité lexicale à base de caractères via un CNN.

L'extraction d'informations morphologiques doit prendre en considération tous les caractères

d'un mot. En arabe, des caractéristiques informatives peuvent apparaître au début (comme la proclitique *particule d'interrogation* "أ" dans "أ يستحق" souvent utilisée dans l'ironie), au milieu (comme la *particule affirmative* "ل" dans "فليقرأه"), ou à la fin.
Des études antérieures [Santos & Zadrozny, 2014; Chiu & Nichols, 2016] ont montré que le CNN est efficace pour extraire des informations morphologiques des caractères des mots. Ils ont utilisé un CNN pour représenter les mots avec des embeddings basés sur les caractères. La figure 5.2 montre le CNN que nous utilisons pour extraire la représentation des caractères d'une unité lexicale donnée. Le CNN est similaire à celui de [Ma & Hovy, 2016]. En effet, les unités lexicales sont initialement enrichies par du padding sur les deux extrémités. Pour chaque unité lexicale, une opération de convolution et un max-pooling sont appliqués pour extraire un nouveau vecteur qui représente l'embedding à base de caractères relatif à l'unité lexicale.

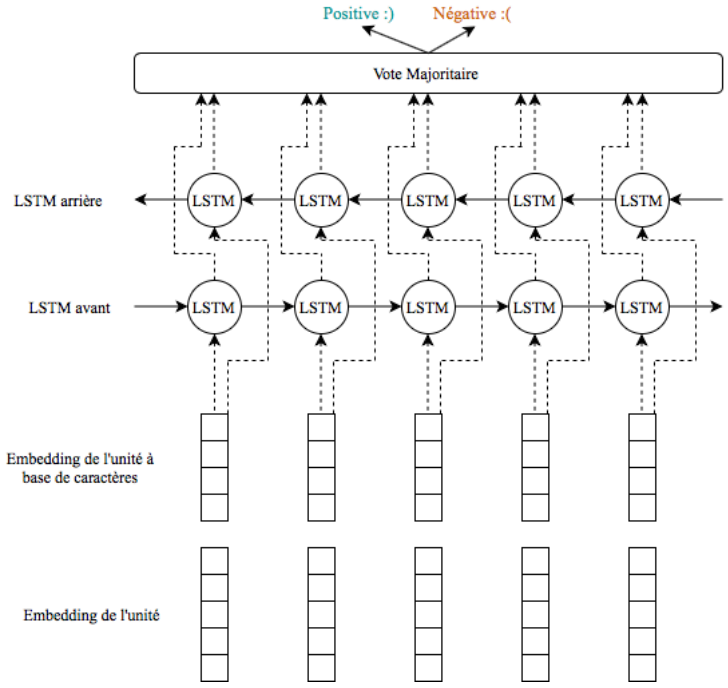


FIGURE 5.3 – Architecture du BiLSTM pour l'analyse d'opinions.

La figure 5.3 illustre l'architecture *BiLSTM* basée sur la boîte à outils *NeuroNLP*[7]. Pour chaque mot du commentaire, l'embedding à base de caractères est obtenu par le CNN (comme illustré dans la figure 5.2). Ensuite, il est concaténé avec des embeddings d'unités pré-entrainés pour alimenter le BiLSTM. Notre système prévoit une classe (positive ou négative) pour chaque unité lexicale du commentaire. La classe faisant référence au nombre maximal d'étiquettes, selon un vote majoritaire, est affectée comme polarité globale du commentaire. Le vote majorité consiste à calculer le nombre de mots classés positivement et le nombre de mots classés négativement. Si une majorité l'emporte, la classe majoritaire est choisie, et dans le cas contraire la polarité positive (classe majoritaire dans le

7. https://github.com/XuezheMax/NeuroNLP2

corpus LABR) est choisie.
Or, la classe négative peut être aussi choisie comme polarité globale du document si pas de majorité. Pour ce décider, nous avons entraînés *NeuroNLP* avec les embeddings de [Dahou *et al.*, 2016], les performances du *BiLSTM* sur l'ensemble de test du corpus LABR étaient très proches, elles sont à 0.03 de différence. Comme la différence est très négligeable, nous choisissons d'attribuer la polarité positive au document dans le cas d'égalité entre le nombre de mots classés positivement et le nombre de mots classés nagativement.

Le CNN permet de calculer les embeddings basés sur les caractères. Ces derniers étaient utiles pour les tâches de reconnaissance d'entitées nommées et d'étiquetage morphosyntaxique [Ma & Hovy, 2016]. Nous décidons donc d'évaluer le *BiLSTM* [8] pour l'AOA.
Dans un premier temps, nous avons commencé par évaluer le *BiLSTM* avec les embeddings de mots de [Dahou *et al.*, 2016]. Pour montrer l'efficacité des embeddings à base de caractères, nous avons testé le *BiLSTM* sans considération des embeddings à base de caractères, et nous avons observé une baisse de 0.2 dans les performances. D'où l'utilité des embeddings à base de caractères dans l'architecture *BiLSTM*.
Par ailleurs, nous avons testé l'architecture CNN présentée dans le chapitre 4 pour l'AOA avec les embeddings à base de caractères. Nous avons observé une baisse de performances de 0.3.
Nous utilisons donc les embeddings à base de caractères uniquement pour l'architecture *BiLSTM*.

Les deux architectures neuronales *CNN* et *BiLSTM* sont utilisées avec les embeddings spécifiques à l'arabe pour prédire la polarité globale des documents (positive ou négative).

5.2.1.1.c Pourcentage de l'ASM dans LABR

Dans le cadre de l'analyse d'opinions en arabe, les internautes ont tendance à exprimer leurs opinions avec différents styles (formel ou informel). Les opinions exprimées peuvent être en ASM, en AD ou mixtes. Pour appliquer les outils TAL appropriés, il est nécessaire d'identifier la langue de l'énoncé. La tâche d'identification de langue s'avère importante afin de valider l'applicabilité de certains outils.

Pour distinguer les mots en ASM des mots en AD et appliquer par la suite les outils TAL adéquats à chaque type d'arabe, nous avons commencé par établir un état de l'art relatif à ce domaine. Dans la littérature, deux approches peuvent être distinguées. La première consiste à considérer l'identification de la langue comme un problème de classification. Quant à la deuxième approche, elle consiste à utiliser des dictionnaires permettant de distinguer l'ASM de l'AD.
La première approche nécessite des corpus annotés qui ne sont pas disponibles. Elle demande des efforts d'annotation qui la rend lente et coûteuse. La deuxième approche semble plus rapide et elle nécessite la disponibilité de dictionnaires. Nous nous situons donc dans le cadre de la deuxième approche. Un mot est considéré en ASM s'il existe dans un dictionnaire d'ASM. Nous présentons, dans la suite, notre processus de construction d'un ensemble de mots en ASM.

8. Nous avons conservé les mêmes paramètres que dans [Ma & Hovy, 2016].

Notre but est de construire le plus grand ensemble de mots en ASM. nous avons utilisé 2 dictionnaires de mots en ASM : *Alwasit* et *Almouaser* contenant respectivement 43 117 et 31 638 mots et leurs définitions. Comme la taille des dictionnaires est petite par rapport au vocabulaire d'ASM, nous avons considéré également les mots dans les définitions afin d'agrandir l'ensemble de mots en ASM. Notre objectif est de dire si un mot est en ASM et non pas de chercher sa définition. Nous avons également utilisé deux livres en ASM : *Tahdhib Allougha* et *Taj Alarous* contenant respectivement 626 444 et 1 469 652 mots. Les deux dictionnaires *Alwasit* et *Almouaser* et les deux livres *Tahdhib Allougha* et *Taj Alarous* sont développés au sein de l'équipe ANLP-RG du laboratoire MIRACL.
Nous avons doublé les mots résultants en appliquant un processus de suppression de voyelles courtes et des signes diacritiques «soukoun» et «chadda». Nous avons appliqué également la lemmatisation sur ces mots afin d'obtenir leurs formes canoniques. L'ensemble global final *dic_final* regroupe 5 585 787 mots arabes différents.

Pour le calcul du pourcentage en ASM de notre corpus LABR, nous avons supposé qu'un mot est en ASM s'il figure dans *dic_final*. Ainsi, 95,78% du LABR est en ASM. Ce pourcentage semble suffisant pour appliquer des outils TAL dédiés à l'ASM.

5.2.1.2 Résultats

Après avoir entraîné word2vec et FastText sur les 6 différentes représentations du corpus *Global*, nous obtenons 12 ensembles d'embeddings. Nous cherchons à comparer leurs performances pour la tâche d'AOA sur le corpus LABR. Rappelons que la meilleure exactitude du système CNN obtenue avec les embeddings de mots pré-entraînés existants est 90.3%. Elle est obtenue avec les embeddings de type FastText nommé *cc.arz*.

Le tableau 5.5 montre les performances de nos architectures CNN et BiLSTM sur l'ensemble de données de test. La meilleure performance est mentionnée en gras et la deuxième meilleure est soulignée. Nous remarquons que quelque soit l'espace d'embeddings, l'exactitude (A) varie entre 88.8% et 91.5%.
Nous constatons que le CNN est plus performant que le BiLSTM, quelle que soit l'unité lexicale. La meilleure performance (91.5%) est obtenue avec le réseau CNN avec des embeddings de lemmes (entraînés avec le modèle word2vec).
Le meilleur système CNN a une précision de 91,5 % et est obtenu avec les embeddings de lemme_w2v, et le second meilleur est obtenu avec les embeddings lemme_FT. Cependant, les deux meilleurs BiLSTM donnent 91 % d'exactitude avec les embeddings de lemme_FT et de lemme_w2v. Nous pourrions déduire que les lemmes représentent la meilleure unité arabe pour l'analyse d'opinions.
De plus, nous notons que CNN et BiLSTM obtiennent à peu près les mêmes performances avec les embeddings de token et token\clitiques. Ceci pourrait expliquer le processus d'élimination des clitiques dans la conscrtuction d'embeddings et justifier la non-nécessité de clitiques pour la tâche d'analyse d'opinions. De plus, nous notons que les embeddings de stemmes donnent, comme prévu,

		CNN				BiLSTM			
Unité	**Emb**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**
mot	w2v	91.1	85.7	76.3	80.7	90.9	83.5	78.3	80.8
	FT	91.2	85.2	77.2	81	90.9	83.1	**79**	**81**
token	w2v	91.2	85.8	76.7	81	90.8	83.8	77.3	80.4
	FT	91.2	85.8	76.7	81	90.7	83.8	76.7	80.1
token\clitiques	w2v	91.2	86.4	76.0	80.9	90.8	83.8	77.2	80.4
	FT	91.2	85.7	76.7	80.9	90.9	**84.6**	76.7	80.4
lemme	w2v	**91.5**	85.8	**78**	**81.7**	**91**	84.3	76.9	80.4
	FT	91.4	**87**	76.4	81.4	**91**	83.6	78.5	**81**
light stemme	w2v	91.2	85.6	76.8	81	90.8	83	78	80.4
	FT	91.4	86.3	76.8	81.3	90.7	83.3	77.4	80.3
stemme	w2v	89	81.2	69.8	75.1	89.3	80.2	73.9	76.9
	FT	88.8	81.3	69	74.6	89.1	79.9	73.4	76.5

TABLE 5.5 – Évaluation de CNN avec les différents embeddings simples.

les faibles résultats car l'espace d'embeddings de stemmes est très petit par rapport aux espaces des autres unités lexicales.

Nous avons calculé la matrice de confusion du meilleur système. Le système prédit bien les critiques positives avec 97,68% de précision et 92,46% de rappel. Les commentaires négatifs sont plus difficiles à détecter avec seulement 54,45% de précision et 81,49% de rappel. Notre meilleur système *CNN + lemme_w2v* est plus performant que le meilleur système testé sur LABR dans le chapitre 4 (le tableau 5.6 rappelle les meilleures performances obtenues dans le chapitre 4). En effet, le *CNN + lemme_w2v* apporte un gain de 1,2 en exactitude (91.5% vs. 90.3%).

Pour expliquer les meilleurs résultats obtenus avec les embeddings d'*ArEmb*, nous effectuons, dans la section 5.2.1.3, une comparaion entre ces embeddings et ceux pré-entrainés exitants.

5.2.1.3 Comparaison avec les embeddings de mots pré-entrainés existants

Nous menons la même analyse d'embddings du chapitre 4 portant sur la couverture et l'édude de voisinage de mots polarisés dans les espaces d'embeddings *ArEmb*.
Le tableau 5.6 rappelle les meilleures performances du CNN obtenues dans le chapitre 4 avec les embeddings mots de type word2vec (w2v) et FastText (FT) dans le cadre de la classification binaire. La couveture de ces embeddings et leurs ratios de positivité et de négativité sont respectivement présentés dans les tableaux 5.7 et 5.8.

5.2.1.3.a Couverture

Les ensembles d'embeddings *ArEmb* couvrent 100% du corpus LABR. Ce taux complet de couverture est dû à l'intégration de l'ensemble d'apprentissage de LABR dans le corpus *Global*. Ceci

Emb		CNN			
Type	**Nom**	**A**	**P**	**R**	**F1**
w2v	wiki-cbow	89.6	83.02	76.12	79.42
FT	cc.arz	**90.3**	85.91	76.02	80.67

TABLE 5.6 – Rappel des meilleures performances du CNN sur le corpus de Test avec les embeddings de mots pré-entraînés existants (de type word2vec et FastText).

Emb		Couverture (%)
Type	**Nom**	
w2v	wiki-cbow	61.53
FT	cc.arz	100

TABLE 5.7 – Rappel des couvertures du corpus LABR par les embeddings de mots pré-entraînés existants : wiki-cbow et *cc.arz*.

		w2v **wiki-cbow**	**FT** ***cc.arz***
$\%^{+}_{Top_n}$	n=2	38.83	100
	n=5	58.75	40.51
	n=10	63.39	30.51
#LABR_lex^{+}		112	235

		w2v **wiki-cbow**	**FT** ***cc.arz***
$\%^{-}_{Top_n}$	n=2	31.77	0
	n=5	12.71	20
	n=10	6.35	40
#LABR_lex^{-}		107	262

TABLE 5.8 – Rappel des ratios de positivité et de négativité des embeddings de mots pré-entraînés existants : wiki-cbow et *cc.arz*.

peut expliquer les meilleures performances obtenues avec *ArEmb*. Cette remarque est valide quelque soit la technique de construction d'embeddings :

— wor2vec (w2v) : le CNN entraîné avec les embeddings de mots d'*ArEmb* donne une exactitude de 91.1% qui est supérieure à celle (89.6%) obtenue avec les embeddings de mots pré-entrainés existants nommé *wiki-cbow* (voir tableau 5.6).

— FatsText (FT) : le CNN entraîné avec les embeddings de mots d'*ArEmb* donne une exactitude de 91.2% qui est supérieure à celle (90.3%) obtenue avec les embeddings de mots pré-entrainés existants nommé *cc.arz* (voir tableau 5.6).

Pour vérifier l'influence de l'intégration de l'ensemble d'apprentissage de LABR dans le corpus d'entrainement d'embeddings *Global*, nous avons entraîné word2vec et FastText sur *Global* sans l'ensemble d'apprentissage de LABR (*Global*$\setminus$*LABR*). Les embeddings word2vec couvrent 99,9% de LABR et FastText le couvre à 100%. Cette couverture élevée peut s'expliquer par le contenu de *Global*$\setminus$*LABR*. Ce dernier contient le corpus BRAD qui ressemble au corpus LABR : les deux partagent le même domaine (critiques sur livres) et donc principalement le même vocabulaire.

Le tableau 5.9 montre le biais de l'utilisation de l'ensemble d'apprentissage LABR sur les performances du système *CNN*[9]. On note un biais de 0,1% avec le modèle word2vec. Cependant, aucun

9. Nous calculons le biais avec le système *CNN* uniquement. Le biais peut être calculé avec le système *BiLSTM*

biais avec FastText. Ceci signifie que l'ensemble d'apprentissage LABR ne joue pas un rôle fondamental dans la qualité de l'espace d'embeddings, et l'utilisation d'un corpus de domaine similaire (ici BRAD) semble suffisant pour assurer de bonnes performances.

	Global	**Global\LABR**
w2v	91.1%	91%
FT	91.2%	91.2%

TABLE 5.9 – Biais de l'utilisation de LABR.

Nous choisissons de considérer, dans la suite, les embeddings entraînés sur le corpus entier *Global*.

5.2.1.3.b Étude de voisinage

Nous étudions, tout d'abord, les ratios de positivité et de négativité, comme défini dans la section 4.4.2 du chapitre 4, des mots polarisés du lexique *LABR_lex* dans i). Nous comparons, ensuite dans ii), ces ratios avec ceux relatifs aux embeddings de mots pré-entrainés existants.

i) Étude des ratios dans l'espace *ArEmb*

Les tableaux 5.10 montrent les ratios de positivité et de négativité des mots polarisés calculés sur le lexique *LABR_lex* dans l'espace des embeddings de mots de *ArEmb*. En ce qui concerne les embeddings de mots construits avec word2vec, nous constatons que plus le voisinage est large, plus le ratio de positivité est petit. Ceci signifie que les mots positifs sont entourés, dans un voisinage restreint, par des mots positifs formant ainsi, dans l'espace d'embeddings, des petites zones composées purement de mots positifs (voir figure 5.4). En revanche, pour le ratio de négativité, nous constatons que plus le voisinage est large, plus le ratio de négativité est relativement grand. Nous remarquons que seulement 4 mots négatifs, en moyenne, se trouvent dans un voisinage de 10. Ceci signifie que les mots négatifs sont plus entourés par des mots positifs, c'est-à-dire les mots négatifs sont éparpillés dans l'espace ne permettant pas de former des zones de mots négatifs. La figure 5.4 explique sous forme graphique les ratios obtenus.

		Mot	
		w2v	**FT**
$\%^{+}_{Top_n}$	n=2	100	100
	n=5	40.17	40.51
	n=10	30.26	30.51
$\#LABR\_lex^{+}$		224	235

		Mot	
		w2v	**FT**
$\%^{-}_{Top_n}$	n=2	0	0
	n=5	20	20
	n=10	40	40
$\#LABR\_lex^{-}$		260	262

TABLE 5.10 – Ratios (en %) de positivité (respectivement de négativité) des mots positifs (respectivement négatifs) dont l'embedding existe à la fois dans *LABR_lex* et l'ensemble d'embeddings de mots d'*ArEmb*.

En ce qui concerne les embeddings de mots construits avec FastText, nous constatons que les ratios de positivité et de négativité des mots polarisés dans l'espace d'embeddings entraînés avec FastText sont proches de ceux dans l'espace d'embeddings entrainés avec word2vec. Cette observation rejoint

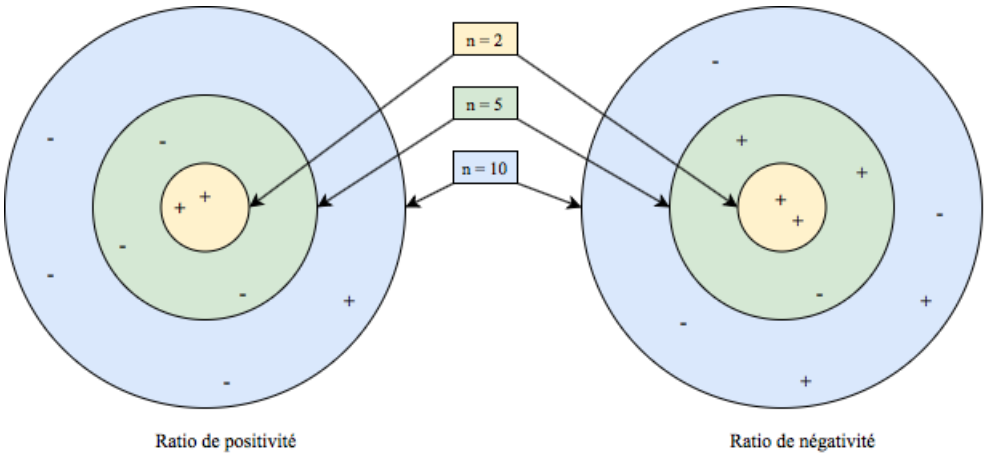


FIGURE 5.4 – Graphiques des ratios de positivité et de négativité des mots polarisés dans l'espace d'embeddings de *ArEmb* entrainés avec word2vec.

les constations de [Bojanowski *et al.*, 2016, 2017] qui portent sur les similarités des espaces d'embeddings construits avec les deux techniques word2vec et FastText. Néamoins, FastText est plus rapide que word2vec. Nous avons remarqué qu'une époque d'entraînement avec word2vec nécessite environ 4 fois le temps pris par une époque d'entraînement avec FastText.

ii) Comparaison des ratios dans les espaces d'embeddings

Nous comparons les ratios des mots polarisés obtenus avec les embeddings de mots d'*ArEmb* (*emb-mot_ArEmb*) et les embeddings pré-entrainés existants. Nous comparons séparement les techniques word2vec et FastText. En ce qui concerne word2vec, nous choisissons la comparaison avec les embeddings de mots pré-entrainés sur wikipédia de [Soliman *et al.*, 2017] (*emb-mot_wiki*) donnant les meilleures performances dans le cadre de la classification binaire [10]. Dans un premier temps, l'espace *emb-mot_wiki* représente 219 mots polarisés du lexique *LABR_lex*, alors que l'espace *emb-mot_ArEmb* permet de représenter plus de mots polarisés (479) du *LABR_lex*. Dans un deuxième temps, et dans un voisinage de 10, les mots positifs sont entourés, en moyenne, par 6 mots positifs dans l'espace *emb-mot_wiki*, et par 3 mots positifs dans l'espace *emb-mot_ArEmb*. Et les mots négatifs sont entourés , en moyenne, par 1 mot négatif dans l'espace *emb-mot_wiki*, et par 4 mots négatifs dans l'espace *emb-mot_ArEmb*. Nous pouvons déduire que les mots positifs sont mieux représentés dans l'espace *emb-mot_wiki* et les mots négatifs sont mieux représentés dans l'espace *emb-mot_ArEmb*. De plus, nous observons un déséquilibre des ratios de positivité (63.39%) et de négativité (6.35%) dans l'espace *emb-mot_wiki*, et nous notons un équilibre des ratios de positivité (30.51%) et de négativité (40%) dans l'espace *emb-mot_ArEmb*.

En ce qui concerne FastText, nous choisissons la comparaison avec les embeddings de mots pré-entrainés *cc.arz* donnant les meilleures performances dans le cadre de la classification binaire. Dans un premier temps, les deux espaces d'embeddings *emb-mot_cc.arz* et *emb-mot_ArEmb* permettent de représenter tous les mots polarisés du *LABR_lex* (487). Dans un deuxième temps, les ratios de

10. Nous rappelons que, dans ce chapitre, nous nous limitons à la classification binaire.

positivité et de négativité obtenus avec l'epace *emb-mot_ArEmb* sont égaux à ceux obtenus avec l'espace *emb-mot_cc.arz*.

5.2.2 Évaluation de combinaison des sorties de systèmes

La combinaison des sorties de systèmes est généralement utilisée pour améliorer les performances. Nous présentons le protocole de combinaison des sorties dans la section 5.2.2.1, et les résultats dans la section 5.2.2.2.

5.2.2.1 Protocole

Une des voies explorées depuis plusieurs années pour améliorer les performances des systèmes d'apprentissage consiste à combiner différents systèmes pour une même tâche. Nous allons nous intéresser ici à une famille de techniques qui font coopérer des systèmes identiques entraînés sur des ensembles d'apprentissage différents. Elles ont été développées vers le milieu des années 90, et ont connu un succès.
Deux protocoles de combinaison sont testés : *oracle* et *consensus*.

- Oracle : il permet de connaître les performances maximales qui pourraient être obtenues avec la combinaison idéale. Il considère l'étiquette correcte si elle est prédite par au moins un système.

- Consensus : il ne prend en compte que les documents associés aux mêmes étiquettes par tous les systèmes. Il mesure leur degré d'accord. En conséquence, la couverture n'est plus de 100 % mais la confiance dans la prédiction est très élevée.

5.2.2.2 Résultats

Nous proposons de combiner les sorties des différents systèmes neuronaux. En fait, ces plongements lexicaux se comportent différemment car ils ont été construits pour différentes unités lexicales arabes et avec différentes techniques. Nous pensons que leur combinaison pourrait améliorer les performances. Les pires performances étaient obtenues avec des embeddings de stemmes, nous les excluons donc du cadre de combinaison.
Les deux protocoles de combinaison *Oracle* et *Consensus* sont testés. Les résultats sont reportés dans le tableau 5.11.

		CNN		**BiLSTM**	
	Protocole	**Couverture**	**Exactitude**	**Couverture**	**Exactitude**
Tous\stemme	*Oracle*	100%	100%	100%	100%
	Consensus	86.57%	100%	81.70%	100%
2 Meilleurs	*Oracle*	100%	92.3%	100%	92.5%
	Consensus	**98.25%**	**92.2%**	96.96%	92.2%

TABLE 5.11 – Évaluation des différents protocoles de combinaison des sorties de systèmes.

Nous testons deux cadres de combinaison. Le premier *Tous\stemme* consiste à combiner tous les systèmes CNN_unit_Emb et BiLSTM_unit_Emb. Nous considérons toutes les unités sauf le stemme. Le deuxième cadre *2 Meilleurs* consiste à combiner les 2 meilleurs systèmes CNN et les 2 meilleurs BiLSTM. Dans *Tous\stemme*, nous obtenons un oracle avec 100% d'exactitude pour CNN et BiLSTM. Cela signifie que la classe de référence est prédite par au moins un système. Nous notons également un accord entre les systèmes sur les avis correctement classés, avec 86,75% de couverture avec CNN et 81,7 % avec BiLSTM.
Dans le cadre *2 Meilleurs*, les systèmes CNN et BiLSTM atteignent la même exactitude 92,2%. Cependant, CNN couvre plus que BiLSTM. La meilleure couverture est de 98,25%. Ainsi, nous concluons que, dans le cadre de combinaison *2 Meilleurs*, CNN définit une *décision forte* plus grande. Les décisions en dehors de ce consensus indiquent des critiques qui doivent être soigneusement analysées.

5.2.2.2.a Analyse du non consensus

Nous analysons le désaccord entre les systèmes CNN, en nous concentrant sur les 146 (1,75% de l'ensemble de test) critiques *non consensus*, du protocole de combinaison *2 Meilleurs*, composées de 68 critiques positives et 78 négatives. Plus précisément, en ce qui concerne les étoiles de classement correspondantes, ces 146 critiques se composent de $39,72\%$ critiques annotées avec 2 étoiles et de $13,69\%$ critiques annotées avec 1 étoile, correspondant à la classe négative de nos systèmes. Concernant la classe positive, les commentaires *non consensus* contiennent $33,56\%$ des critiques annotées avec 4 étoiles et seulement $13,01\%$ des critiques annotées avec 5 étoiles.
Une première analyse du contenu des critiques nous a mené aux remarques suivantes :

- **Critique mixte :** l'internaute peut exprimer une opinion purement positive ou négative. Comme, il peut exprimer une opinion mixte, à la fois postive et négative. Cette mixture de polarité est généralement faite avec des termes de concession (لكن (mais), مع ذلك (cependant), غير ان (toutefois), *etc*). Grâce à ces termes d'opposition, l'internaute peut exprimer, dans la même critique, une opinion positive et une autre négative : l'une dans la partie précédant le terme de concession et l'autre dans la partie restante. En revanche, il peut associer à sa critique un nombre d'étoiles portant uniquement sur l'une des deux parties.
- **Nombre d'étoiles :** dans certains cas, l'internaute mentionne le nombre d'étoiles qu'il va associer à son commentaire.
- **Évaluation de l'auteur :** c'est le cas où l'internaute évalue l'auteur et pas le livre.

En plus de cette analyse de contenu des critiques, nous avons ré-annoté les 278 reviews non-consensus de l'ensemble de Dev par 3 annotateurs humains natifs arabes. Cette annotation comporte deux volets. Le premier consiste à attribuer les labels "positive" ou "négative". Le deuxième consiste à annoter en étoiles $\in \{1,2,3,4,5\}$: c'est le même jeu d'étiquettes utilisé par l'internaute pour classer son commentaire.

	Exemples	
Critique mixte	المواضيع تقليدية لكن الاسلوب جميل	Sujets traditionnels mais le style est beau
	اسلوب قوي واحداث ضعيفة	Style fort et événements faibles
Évaluation d'auteur	على علي زيدان ان يترك الكتابة ليبقى كبيرا بعيون الناس	Aly Zaydan Il doit quitter l'écriture pour rester grand aux yeux des gens
	لا تعليق، فاي تعليق يقلّل من قيمة الكاتب	Aucun commentaire, tout commentaire réduit la valeur de l'auteur
Nombre d'étoiles	لقد كنت كريمة جدا باعطاء نجمتين	J'ai été très généreux en donnant deux étoiles
	انقصت نجمة في التقيم	Une étoile de moins dans l'évaluation

Dans le cadre du premier volet, l'accord inter-annotateur vaut 0.64, ce qui signifie que la concordance est important mais pas parfaite. Parmi les 278 reviews non-consensus, les annotateurs n'étaient pas d'accords sur 27% (75) des reviews. Il y avait un accord sur les 203 commentaires restants. Parmi ces 203 reviews, seulement 18 commentaires ne correspondent pas à la référence, c'est-à-dire la polarité associée à ces 18 commentaires par les 3 annotateurs est différente de celle attribuée par le détenteur de l'opinion (la personne qui a commenté).
Sur le deuxième volet, l'accord inter-annotateur est égal à 0.22. Ce qui signifie une très légère concordance. En effet, il y avait un accord sur 36.33% des reviews et un désaccord sur le reste. De plus, au moins 33.45% des reviews paraissent ambigues pour un des annotateurs.

5.2.2.2.b Analyse du consensus

Les systèmes sont d'accords sur 4 722 reviews du corpus de validation *Dev*. Parmi les quelles, 304 étaient mal-classées (57 positives et 247 négatives) : ce qui représente 6.43% du consensus avec une grande proportion du négatif (81.25%). Nous avons choisi aléatoirement 50 reviews parmi les négatives mal-classées et nous avons appliqué le même processus d'annotation décrit ci-dessus. Nous avons remarqué que 20% des reviews paraissent ambigues dont 26% sont positives et 34% sont négatives. Cette constatation nous conduit à poser la question sur la fiabilité des données et surtout celles d'apprentissage, ce qui pourrait impacter la qualité du modèle d'apprentissage.

5.2.2.2.c Fiabilité des données d'apprentissage

La qualité du modèle de classification est assurée par la qualité des données utilisées lors de l'apprentissage. Plus la qualité des données est bonne, plus le modèle est performant dans la prédiction et la généralisation. L'analyse menée en 5.2.2.2.a et pourrait expliquer les performances des systèmes par la qualité du modèle d'apprentissage. Dans cette perspective, nous décidons de choisir convenablement les données qui vont participer à l'apprentissage des modèles. Pour ce faire, nous avons divisé le corpus de train de LABR en 5 ensembles : chacun contenant 877 reviews négatives et 4762 positives. Une validation croisée de 5 folds a été appliquée sur le train initial : on sélectionne un des

5 échantillons comme ensemble de test et les autres échantillons constituent l'ensemble d'apprentissage. Ce processus a été appliqué pour les 2 systèmes CNN_lemme_w2v, CNN_lemme_FT. Nous récupérons les reviews consensus de chaque fold pour former l'ensemble *train_sélection*. Cet ensemble *train_sélection* pourrait être condidéré plus fiable puisqu'il y avait consensus sur les exemples qui le composent. Nous obtenons ainsi le nouveau corpus d'apprentissage *train_sélection* plus fiable mais plus petit. La taille du corpus *train_sélection*, égale à 21 674, est plus petite que celle du corpus *train* initial qui vaut 36 688.
Les systèmes sont re-entraînés sur le corpus *train_sélection*. Leurs performances sont légèrement moins bonnes que celles obtenues avec le corpus d'apprentissage original. Nous avons calculé, par la suite, leur consensus *consensus_sélection* et obtenu 93.5% d'exactitude sur 94.5% du corpus de validation *Dev*.
Cette performance est équivalente à celle obtenue avec le consensus *consensus_total* entre les 2 systèmes entraînés sur le corpus d'apprentissage initial. Le processus de sélection de données semble donc avoir la même qualité du modèle d'apprentissage que sans sélection de données. Dans le corpus de validation , les reviews consensus sont au nombre de 4 723 (respectivement 4 722) avec le protocole *consensus_sélection* (respectivement *consensus_total*). En outre, nous avons identifié les reviews retenues par les deux consensus. Ces dernières sont au nombre 4 582 reviews : 4 031 positives et 551 négatives.

5.2.3 Évaluation de combinaison des entrées de systèmes

En plus de la combinaison des sorties de systèmes, il est possible de combiner différentes entrées afin d'améliorer les performances. Nous présentons le protocole de combinaison des entrées dans la section 5.2.3.1 et les résultats dans la section 5.2.3.2.

5.2.3.1 Protocole

Les entrées de nos systèmes neuronaux sont les différents embeddings d'unités lexicales *ArEmb*. Nous testons trois approches de combinaison d'embeddings : concaténation simple, analyse en composantes principales et auto-encodeur [Ghannay *et al.*, 2016].

- Concaténation simple

La première approche, appelée *Concat* dans ce qui suit, consiste à concaténer les différents embeddings. L'embedding résultant est représenté par un vecteur de dimension d qui est égale à la somme des dimensions des embeddings concaténés.

- Analyse en composantes principales

La deuxième approche *ACP* consiste à combiner les embeddings par application d'une méthode d'analyse en composantes principales qui est une méthode classique en analyse de données. Son objectif est de réduire la dimension de l'espace tout en gardant le plus possible les caractéristiques

pertinentes de l'espace. Il s'agit d'obtenir le résumé le plus pertinent des données initiales. Mathématiquement, l'ACP est définie par un simple changement de base. Elle consiste à passer d'une représentation dans la base des variables initiales à une représentation dans la base des facteurs définis par les vecteurs propres de la matrice de corrélation.
En pratique, l'ACP est appliquée sur l'embedding obtenu par concaténation afin de réduire sa dimension.

- Auto-Encodeur

La dernière approche *AE* porte sur l'utilisation d'un auto-encodeur pour la combinaison des embeddings. L'auto-encodeur est une généralisation non linéaire de l'ACP. Un encodeur est utilisé pour transformer des données d'une représentation à grande dimension en une représentation de faible dimension, puis un décodeur est utilisé pour reconstruire les données à partir de cette représentation.
Pratiquement, nous avons appliqué l'auto-encodeur sur l'embedding obtenu par concaténation afin de réduire sa dimension. La représentation résultante de l'*AE* est susceptible de préserver les informations utiles et non redondantes de la représentation initiale.

5.2.3.2 Résultats

Dans cette section, nous reportons les résultats d'évaluation des embeddings combinés avec les différentes approches de combinaison présentées dans la section 5.2.3.1.
Comme les meileures performances des CNN et BiLSTM sont obtenues avec les lemmes, nous choisissons d'appliquer les approches de combinaison d'embeddings sur l'unité lexicale *lemme* qui représente une forme réduite de mot. Nous décidons également d'appliquer ces approches sur l'unité lexicale *mot* qui constitue la composition par défault de textes arabes. Les tableaux 5.12 et 5.13 rapportent les performances du CNN sur le corpus de Test avec les embeddings combinés aux niveaux des unités lexicales *lemme* et *mot* repectivement. La partie grisée représente les caractéristiques de la combinaison donnant les meilleures performances sur le corpus de Dev. Les résultats de combinaion sur le corpus de Dev sont présentés dans les tableaux A.2 et A.3 de l'annexe A.

En ce qui concerne l'unité *lemme*, les résultats de combinaison sont reportés dans le tableau 5.12. La meilleure performance (91.4%) est obtenue avec les embeddings de lemmes (de dimension 400) obtenus avec la combinaison *ACP* d'embeddings de lemmes entraînés avec w2v et FT. Cette performance ne dépasse pas celle obtenue avec les embeddings de lemmes entraînés avec w2v (91.5%) et avec les embeddings de lemmes entraînés avec FT (91.4%). Nous concluons que la combinaison au niveau de l'unité lexicale *lemme* n'améliore pas les performances du CNN.

En ce qui concerne l'unité *mot*, les résultats de combinaison sont reportés dans le tableau 5.13. La meilleure performance (90.9%) est obtenue avec les embeddings de mots (de dimension 400) obtenus avec la combinaison *ACP* d'embeddings de mots entraînés avec w2v et FT. Cette performance est inférieure à celle obtenue avec les embeddings de mots entraînés avec w2v (91.1%) et avec les embeddings de mots entraînés avec FT (91.2%). La combinaison au niveau de l'unité lexicale *mot*

Embeddings	Dimension	A	P	R	F1
Concat	600	91	84.01	78.40	81.11
ACP	400	**91.4**	85.17	78.61	**81.76**
	300	91.2	85.39	77.05	81.00
	200	91.1	85.39	76.79	80.86
	100	90.6	84.72	74.75	79.42
AutoEnc	400	90.30	82.86	75.62	79.08
	300	90.38	84.51	73.78	78.78
	200	90.10	83.31	73.71	78.22
	100	90.32	82.73	76.00	79.23

TABLE 5.12 – Évaluation des différents protocoles de combinaison d'embeddings de lemmes sur le Test avec le système CNN.

Embeddings	Dimension	A	P	R	F1
Concat	600	90.8	87.62	72.76	79.50
ACP	400	**90.9**	85.38	75.97	**80.40**
	300	90.7	84.89	75.37	79.85
	200	90.6	85.31	74.12	79.32
	100	90	84.47	72.25	77.88
AutoEnc	400	89.9	83.55	72.79	77.80
	300	89.78	83.31	71.94	77.21
	200	89.81	82.99	72.48	77.38
	100	89.93	83.33	72.72	77.67

TABLE 5.13 – Évaluation des différents protocoles de combinaison d'embeddings de mots sur le Test avec le système CNN.

n'améliore pas les performances. Nous comparons maintenant les résultats de combinaison aux niveaux lemme et mot. Quelque soit l'approche de combinaison, les performances au niveau lemme sont supérieures à celles au niveau mot. Nous rejoignons ainsi la conclusion portant sur la compléxité de l'unité lexicale *mot* et l'importance de considérer une représentation surfacique réduite du *mot* (ici la représentation *lemme*).

5.3 Analyse qualitative des embeddings spécifiques à l'arabe

Le but de cette section est de mener une analyse qualitative des embeddings spécifiques à l'arabe présentés dans la section 5.2.1. L'analyse se fonde sur des méthodes d'évaluation d'embeddings. Les techniques d'évaluation d'embeddings se divisent en deux catégories : intrinsèque et extrinsèque. D'une part, les méthodes intrinsèques d'évaluation [Baroni *et al.*, 2014; Schnabel *et al.*, 2015] consistent à quantifier directement différentes régularités linguistiques dans l'espace d'embeddings. Les analogies syntaxiques et sémantiques [Mikolov *et al.*, 2013b; Nayak *et al.*, 2016] sont les méthodes intrinsèques les plus utilisées. D'un autre côté, les méthodes d'évaluation extrinsèque évaluent

la qualité des embeddings pour d'autres tâches TAL telles que l'étiquetage morpho-syntaxique, la segmentation, la reconnaissance d'entités nommées, l'analyse d'opinions, *etc*.
La majorité des travaux traitant de l'évaluation des embeddings de mots arabes utilisent une technique extrinsèque. De nombreuses techniques de construction d'embeddings en arabe ont été évalués dans des applications telles que la traduction automatique [Shapiro & Duh, 2018; Lachraf *et al.*, 2019], l'analyse d'opinions [Dahou *et al.*, 2016; Soliman *et al.*, 2017; Fouad *et al.*, 2019], la recherche d'informations [El Mahdaouy *et al.*, 2018], *etc*. De nombreuses applications en aval montrent l'utilité des embeddings de mots. Pour l'évaluation intrinsèque des embeddings de mots arabes, [Elrazzaz *et al.*, 2017] est le seul travail à notre connaissance. Il quantifie les analogies syntaxiques et sémantiques dans les espaces d'embeddings.

Notre objectif est d'évaluer nos embeddings spécifiques à l'arabe pour l'analyse d'opinions en arabe. Nous rappelons que ces embeddings ont été construits à partir du corpus *Global*. Ce dernier contient des corpus dédiés à la tâche d'analyse d'opinions et d'autres génériques. Nous menons une analyse fine dont la méthodologie est détaillée dans la section 5.3.1. Nous présentons et discutons les résultats dans les sections 5.3.2 et 5.3.3 respectivement.

5.3.1 Méthodologie

Les techniques de construction d'embeddings sont entraînées avec plusieurs paramètres : taille des embeddings, taille de la fenêtre, le corpus d'entraînement, le nombre d'époques, *etc*.
[Antoniak & Mimno, 2018] a constaté que les voisins les plus proches sont très sensibles aux petits changements dans le corpus d'entraînement des embeddings. [Pierrejean & Tanguy, 2018] a exploré l'impact du changement de la taille de la fenêtre, de la dimension d'embeddings et de la nuture des corpus d'entraînement. Dans ce chapitre, nous voulons savoir si le type de corpus d'entraînement affecte la performance pour la tâche d'AOA. En d'autres termes, est-il préférable d'entrainer des embeddings avec des corpus (polarisés) spécifiques à la tâche d'AO ? Ou les corpus génériques sont plus efficaces ? Et quelle est la taille adéquate du corpus ?

Pour une étude rigoureuse, nous construisons des espaces d'embeddings entraînés avec trois types de corpus : polarisé, non polarisé et mixte au niveau de mots et de lemmes. Ceci sera détaillé dans la section 5.3.1.3. En plus du type de corpus, nous étudions l'impact du nombre d'époques utilisé pour l'entraînement des embeddings sur le voisinage des unités polarisés.

5.3.1.1 Méthode d'évaluation intrinsèque

Pour l'évaluation intrinsèque des embeddings, nous n'avons pas utilisé de méthodes habituelles basées sur des analogies syntaxiques et sémantiques qui sont génériques et non spécifiques à la tâche d'AO. En considérant le cadre de l'AO, nous introduisons la notion de la stabilité de polarités *sentiment stability (SS)* des unités polarisés (mot et lemme) dans les espaces d'embeddings. Nous supposons que les unités polarisées dans les espaces d'embeddings sont entourées d'unités polarisées. Pour

chaque unité polarisée dans ArSentLex, nous étudions leurs voisins les plus proches afin de voir s'ils se trouvent également dans le lexique ArSentLex (Lex_mot et Lex_lemme).

Afin d'analyser la SS de chaque modèle à une époque spécifique, nous considérons en même temps : une unité lexicale $\in \{$mot, lemme$\}$, un type de corpus $\in \{$ polarisé, non polarisé, mixte$\}$ et un nombre d'époque, et nous calculons combien d'unités positives (respectivement négatives) sont dans les voisins les plus proches pour des unités positives (et respectivement pour les unités négatives). Les voisins les plus proches d'une unité lexicale donnée sont des unités ayant le score de similarité cosinus le plus élevé.

Quelle que soit l'unité lexicale, la stabilité du sentiment consiste à étudier la stabilité du *ratio de positivité* $(\%^{+}_{Top_n})_{ep_i}$ et du ratio de négativité $(\%^{-}_{Top_n})_{ep_i}$ selon les nombres d'époque. Nous avons défini les voisins les plus proches par $n = 10$ qui est la valeur la plus utilisée dans la littérature.
Le rapport de positivité à l'époque ep_i est calculé pour les unités positives (en lexique positif $lexicon^{+}$) avec la formule (5.1) :

$$(\%^{+}_{Top_n})_{ep_i} = 100 \times \frac{\sum_{unit_i \in \{lexique^{+}\}} \#unit^{lexique^{+}}_{Top_n}}{n \times \#lexique^{+}} \tag{5.1}$$

avec : n le nombre des unités les plus proches, $\#unit^{lexique^{+}}_{Top_n}$ le nombre d'unité positives dans les voisins les plus proches top_n de l'$unit$, $\#lexique^{+}$ la taille du lexique polarisé positivement.

Nous calculons également *ratio de négativité* selon la même formule (5.1) en considérant uniquement les unités négatives (dans le lexique négatif $lexique^{-}$).

5.3.1.2 Méthode d'évaluation extrinsèque

Nous entraînons et évaluons le réseau convolutif CNN, dans sa version statique et non statique, avec tous nos embeddings construits en tenant compte de tous les types de corpus {polarisé, non polarisé, mixte}, les unités {mot, lemme} et les numéros d'époques {1, 5, 10, 15, 20}.

5.3.1.3 Espaces d'embeddings

Pour la construction des embeddings, nous considérons trois types de corpus : polarisé, non polarisé et mixte. Les corpus polarisés sont des des ensembles de documents utilisés dans l'AOA. Ils contiennent des avis classés par étoiles rédigés par des internautes afin d'exprimer leurs opinions concernant une entité donnée. Ces avis sont annotés en étoiles sur une échelle de 1 à 5 étoiles. Notre corpus polarisé est une concaténation de BRAD [Elnagar & Einea, 2018], HARD [Elnagar *et al.*, 2018a] et l'ensemble d'apprentissage du LABR [Nabil *et al.*, 2014].
Le corpus non polarisé ne contient pas d'avis notés. Il est composé d'énoncés textuels génériques qui ne sont pas spécifiques à l'AOA. Nous choisissons AbuElKhair [El-Khair, 2016] (contenant des articles de journaux) pour construire le corpus non polarisé.
Pour aller plus loin, et afin d'avoir un corpus plus grand, nous concaténons les corpus polarisé et non polarisé précédents et obtenons un corpus mixte.

Un prétraitement est appliqué pour nettoyer et normaliser nos trois corpus. Nous avons appliqué le même prétraitement détaillé dans le chapitre 5. Ensuite, l'outil de lemmatisation Farasa [11] a été appliqué afin de lemmatiser nos trois corpus. Par conséquent, nous obtenons deux ensembles de données par type de corpus : chacun correspond à une unité lexicale $\in \{\text{mot,lemme}\}$. La taille de chaque ensemble de données est indiquée dans le tableau 5.14. Comme prévu, il y a moins de lemmes que de mots. En effet, la taille du vocabulaire de la lemme représente environ la moitié de la taille du vocabulaire des mots. De plus, le nombre de lemmes apparaissant une seule fois est très faible par rapport au nombre de mots apparaissant une fois dans le corpus. Ainsi, la lemmatisation aide réellement à réduire l'aspect épars du vocabulaire arabe.

Type	**Mot**			**Lemme**		
	#voc	**#occ**	**#1**	**#voc**	**#occ**	**#1**
Polarisé	897	47 932	454	392	46 809	245
Non polarisé	2 875	1 190 762	1231	1 392	1 185 847	726
Mixte	3 196	1 238 695	1441	1 614	1 232 656	885

TABLE 5.14 – Statistiques par type de corpus, avec #voc : taille du vocabulaire, #occ : taille du corpus, et #1 : nombre des unités apparaissant une seule fois dans le corpus. Toutes les tailles sont reportées en *kilo (k)*.

5.3.2 Résultats

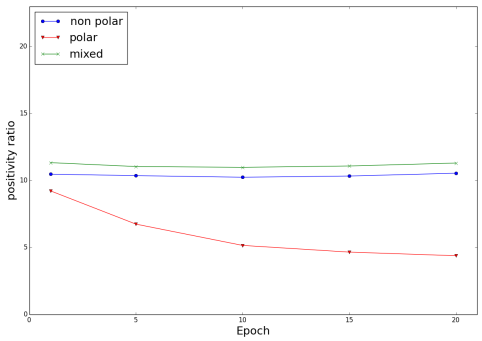
Nous présentons les résultats des évaluation intrinsèque et extrinsèque respectivement dans les sections 5.3.2.1 et 5.3.2.2. Nous discutons les résultats dans la section 5.3.3.

L'unité lexicale *lemme* représente l'unité la plus adéquate à l'analyse d'opinions en arabe. Or, le mot représente l'unité lexicale de base des textes arabes. Nous choisissons donc de mener notre analyse qualitative sur les embeddings de mots et de lemmes. Les meilleures performances sont obtenues avec les embeddings de lemmes de type word2vec. Nous limitons donc notre étude aux embeddings de type word2vec.

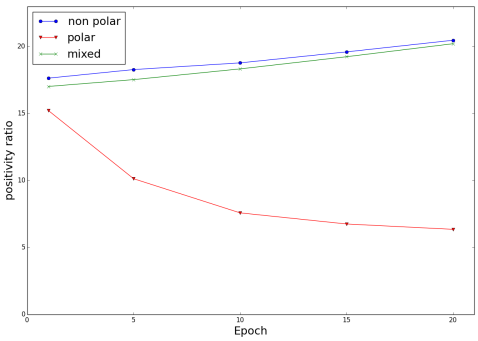
5.3.2.1 Résultats de l'évaluation intrinsèque

Les résultats sont rapportés dans la figure 5.5. Notons que, plus la différence entre les ratios de positivité (ou de négativité) est faible, plus la stabilité du sentiment est élevée. Cela signifie que le modèle est plus stable lorsqu'il y a moins de différence entre les rapports de positivité (ou de négativité) le long des époques. Nous remarquons que les ratios de positivité et de négativité $(\%^{+/-}_{Top_n})_{ep_i}$ sont proches quel que soit le nombre d'époques $i \in \{1,5,10,15,20\}$ pour les corpus non polarisés et mixtes. Nous observons également que ces modèles sont plus stables que ceux obtenus avec des corpus polarisés.

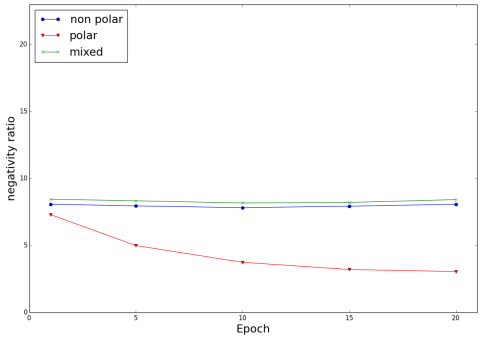
11. http ://qatsdemo.cloudapp.net/farasa/



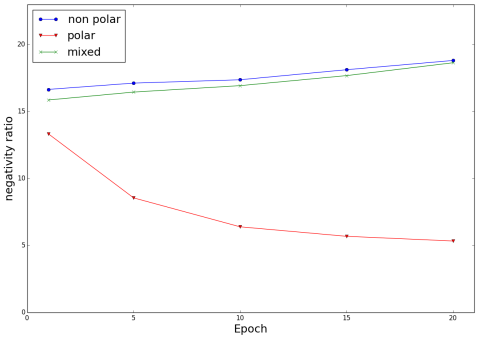
(A) Ratio de positivité au niveau mot



(B) Ratio de positivité au niveau lemme



(C) Ratio de négativité au niveau mot



(D) Ratio de négativité au niveau lemme

FIGURE 5.5 – *Sentiment stability* des embeddings de mots et de lemmes en fonction du type de corpus.

Les histogrammes de la figure 5.5 montrent qu'il y a, au maximum, 3 points de différence de rapport dans les modèles entraînés avec des corpus non polaires et mixtes. Contrairement au corpus polarisé, il y a au moins 3 points de différence de rapport. Nous pourrions conclure que les modèles non polarisés et mixtes sont plus stables que les modèles polarisés. Néanmoins, nous ne devons pas ignorer la taille relativement petite des corpus polarisés (voir tableau 5.14) qui pourrait probablement expliquer la non stabilité du modèle correspondant.
Pour les modèles polarisés, on note que les meilleurs ratios sont obtenus avec des modèles entrainés à l'époque 1. Ensuite, les ratios diminuent le long des époques. Il semble, pour plus d'époques, que les unités polarisées soient moins proches et plus dispersées dans les espaces d'embeddings.

Une comparaison entre les ratios de mots et ceux de lemmes représente un point pertinent dans ce chapitre. Nous remarquons que les rapports de positivité et de négativité des modèles de lemme sont supérieurs à ceux des modèles de mots, quel que soit le nombre d'époques et pour tous les types de corpus d'apprentissage. Cela pourrait justifier notre hypothèse initiale concernant la nécessité de prendre en compte la spécificité arabe. Et cela pourrait prouver l'utilité des embeddings de lemmes pour la tâche d'analyse d'opinions en arabe.

5.3.2.2 Résultats de l'évaluation extrinsèque

Le tableau 5.15 rapporte les performances du CNN entraînés avec les embeddings proposés. Nous ne mentionnons que le numéro d'époque ep_i du modèle d'embeddings donnant la meilleure exactitude (A).

Embeddings		**CNN**			
		statique		**non statique**	
Corpus	Unité	A	ep_i	A	ep_i
polarisé	mot	91.5	1	91.2	20
	lemme	**91.9**	1	**91.6**	1
non polarisé	mot	90.9	10	90.9	1
	lemme	91.0	10	91.3	15
mixte	mot	91.2	15	91.0	1
	lemme	91.3	1	91.1	15

TABLE 5.15 – Exactitude (A) du système CNN entrainé avec les embeddings polarisés, non polarisés and mixtes sur le corpus de test.

Notre meilleur système est obtenu avec la version statique du CNN et les embeddings de lemmes pré-entrainés avec un corpus polarisé. Sur la base de sa matrice de confusion, notre meilleur système prédit des critiques positives avec une précision de 93.06% et un rappel de 97.76%. Les revues négatives sont plus difficiles à détecter avec une précision de 82.01% et seulement 58.46% de rappel. Notre système montre donc une faiblesse dans la prédiction de classe négative.
Notre meilleur système donne une exactitude de 91.9%. Cette performance dépasse les systèmes existants testés sur le corpus LABR. Or dans le chapitre 5, la meilleure exactitude (91.5%) est obtenue avec un CNN non statique. Ce qui signifie que notre système a un gain absolu de 0.4.

Il est également important de noter que les performances obtenues avec les plongements de lemmes sont légèrement supérieures à celles obtenues avec les plongements de mots. Cela est vrai pour les deux versions du CNN statique et non statique, et quel que soit le type de corpus utilisé pour l'entraînement des embeddings. Il semble que les embeddings de lemmes soient meilleurs pour la tâche d'AOA. La taille des ensembles d'embeddings de lemmes représente environ la moitié de la taille des ensembles d'intégration de mots (voir tableau 5.14). Cela signifie que nous pourrions obtenir des performances compétitives avec seulement la moitié de taille de l'ensemble d'embeddings.

De plus, nous observons que pour le CNN statique ou non statique, les meilleurs résultats sont obtenus avec des plongements pré-entrainés avec des corpus polarisés. Nous pourrions conclure que, pour la tâche d'AO, il est assez bon d'utiliser des modèles d'embeddings pré-entrainés avec des corpus spécifiques à la tâche (même petits) que des corpus génériques (de taille plus grande).
De plus, afin de comparer les versions du CNN statique et non statique selon les types de corpus, on remarque que pour les plongements polarisés, les meilleurs résultats sont obtenus avec le CNN statique. Cependant, pour les plongements non polarisés, les meilleurs résultats sont obtenus avec le

CNN non statique. Nous pourrions conclure que pour CNN statique, il est préférable d'utiliser des plongements pré-entrainés avec des corpus spécifiques à la tâche. Cependant, lorsque les plongements sont entraînés avec des corpus non spécifiques à une tâche, il est préférable d'utiliser un CNN non statique. Il semble que pour une tâche d'AO, il est préférable d'utiliser des embeddings pré-entraînés avec des corpus spécifiques à la tâche plutôt que des embeddings spécifiques à la tâche.

Avec les modèles polarisés, les meilleurs résultats sont obtenus avec le CNN statique. Dans le cadre CNN statique, nous notons que les espaces d'embeddings pré-entrainés à l'époque 1 (ep_1) donnent les meilleures performances. Ces résultats peuvent être justifiés par la qualité des espaces d'embeddings mentionnés dans la section 5.3.2.1. En effet, si nous joignons des performances du CNN statique avec des ratios de positivité et de négativité de modèles d'embeddings (détaillés dans la section 5.3.2.1), nous pourrions conclure que dans le CNN statique, plus les ratios sont élevés, meilleures sont les performances du CNN.

5.3.3 Discussion

Nous disposons de deux natures d'embeddings selon la version du CNN :

— embeddings hors tâche d'application obtenus avec la version statique du CNN

— embeddings spécifiques à la tâche d'application obtenus avec la version non statique du CNN

Notre objectif est d'analyser l'évolution dans les espaces d'embeddings. Pour ce faire, nous mesurons les déplacements des positions de mots dans l'espace d'embeddings avant et après entraînement du CNN non statique.

Dans cette section, nous étudions les versions statique et non statique du réseau convolutif. L'objectif est d'expliquer les performances du CNN et comprendre le gain obtenu avec la version statique.

Nous nons focalisons aux embeddings pré-entrainés avec le corpus polarisé donnant, au niveau mot et lemme, les meilleures performances du CNN dans ses versions statique et non statique (voir tableau 5.15).

Pour comprendre le gain obtenu avec la version statique du CNN, nous analysons les positions des unités lexicales (mots et lemmes) participant dans l'apprentissage du CNN et la prédiction de polarité (positive ou négative).

Le tableau 5.16 montre la différence dans les espaces d'embeddings de mots entraînés avec 1 époque et 20 époques avec et sans mise à jour lors de l'apprentissage du CNN. Pour les embeddings non mis à jours, nous observons qu'il n'y a pas de grandes différences en les embeddings obtenus après 1 époque ou après 20 époques. Un mot change d'une position à une autre avec, au grand maximum, une différence de similarité cosinus de 0.3. Cependant, avec les embeddings mis à jour lors de l'apprentissage du CNN (version non statique), un grand changement de position de mots est remarquable. En effet, la majorité de mots a changé de position avec une différence de similarité cosinus (entre l'ancienne et la nouvelle position) comprise entre 0.6 et 1.

Le tableau 5.17 montre la différence dans les espaces d'embeddings de lemmes entraînés avec l'époque 1 avec et sans mise à jour lors de l'apprentissage du CNN. Lors de l'apprentissage du CNN,

Similarité	Mot							
	non màj (ep1) vs. non màj (ep20)				non màj (ep20) vs. màj (ep20)			
	#	#+	#−	#±	#	#+	#−	#±
[-1, -0.1[	0	0	0	0	0	0	0	0
[-1, 0[	0	0	0	0	491	94	90	25
[0, 0.1[	0	0	0	0	10 726	2 030	1 760	484
[0.1, 0.2[	0	0	0	0	15 245	3 336	2 686	798
[0.2, 0.3[	0	0	0	0	2 513	686	468	177
[0.3, 0.4[	0	0	0	0	240	90	57	26
[0.4, 0.5[	0	0	0	0	40	14	11	1
[0.5, 0.6[	0	0	0	0	8	2	3	0
[0.6, 0.7[	9	1	2	1	1	0	0	0
[0.7, 0.8[	8 565	1 166	1 106	306	0	0	0	0
[0.8, 0.9[	20 748	5 228	4 037	1 293	0	0	0	0
[0.9, 1]	672	282	189	73	560	316	195	119

TABLE 5.16 – Statistiques sur les espaces d'embeddings de mots utilisés dans les versions statique (non mis à jour *màj*) et non statique du CNN (mis à jour *màj*), avec : # : le nombre de mots bougés dans l'espace, #+ : le nombre de mots polarisés positivement bougés, #− : le nombre de mots polarisés négativement bougés, et #± : le nombre de mots bougés qui sont positifs et négatifs à la fois

Similarité	Lemme			
	non màj (ep1) vs. màj (ep1)			
	#	#+	#−	#±
[-1, -0.1[	0	0	0	0
[-0.1, 0[	10	3	3	1
[0, 0.1[	524	297	285	183
[0.1, 0.2[	5 200	2 867	2 585	1 585
[0.2, 0.3[	8 889	4 257	3 932	2 320
[0.3, 0.4[	5 980	1 963	1 817	928
[0.4, 0.5[	3 528	717	623	278
[0.5, 0.6[	1 895	217	197	82
[0.6, 0.7[	1 154	65	56	24
[0.7, 0.8[	843	15	25	7
[0.8, 0.9[	96	1	2	0
[0.9, 1]	799	701	600	548

TABLE 5.17 – Statistiques sur les espaces d'embeddings de lemmes utilisés dans les versions statique (non mis à jour *màj*) et non statique (mis à jour *màj*) du CNN, avec : # : le nombre de lemmes bougés dans l'espace, #+ : le nombre de lemmes polarisés positivement bougés, #− : le nombre de lemmes polarisés négativement bougés, et # ± : le nombre de lemmes bougés qui sont positifs et négatifs à la fois

un grand changement de position de mots est remarquable. En effet, la majorité de mots a changé de position avec une différence de similarité cosinus (entre l'ancienne et la nouvelle position) comprise entre 0.4 et 1.

5.4 Conclusion

Les espaces d'embeddings pré-entrainés existants représentent un mot arabe sans considération de l'agglutination et de la richesse morphologique de l'arabe. Dans ce chapitre, nous avons proposé des embeddings prenant en compte les spécificités de la langue arabe. Nous avons défini les différentes représentations possibles d'un mot obtenant ainsi 6 unités lexicales : mot, token, token\clitiques, lemme, light stemme et stemme. Nous avons entrainé, pour chaque unité lexicale, word2vec et FastText pour obtenir à la fin 12 espaces d'embeddings. Ces derniers ont été évalués pour l'analyse d'opinions en arabe (AOA).

Pour choisir l'unité lexicale la plus adéquate pour l'AOA, nous avons testé deux architectures neuronales de nature différente : réseau convolutif (version non statique) et réseau récurrent de type BiLSTM. Les meilleures performances du CNN et BiLSTM sont obtenues au niveau lemme. Il semble que l'unité lexicale *lemme* soit la plus adéquate pour l'AOA.
Quelque soit l'unité lexicale, le CNN est plus performant que le BiLSTM. Nous avons testé des approches de combinaison d'embeddings (concaténation, analyse de composantes principales (ACP) et auto-encodeur). Les performances du CNN obtenues avec les embeddings combinés avec ACP sont meilleures que celles obtenues avec les embeddings combinés avec les autres approches (concaténation et auto-encodeur). Néamoins, pas de gain : les embeddings combinés avec ACP n'améliorent pas les performance des systèmes

Finalement, nous avons étudié la qualité des embeddings spécifiques à l'arabe. Deux méthodes d'évaluation d'embeddings ont été testées : une méthode intrinsèque et une autre extrinsèque. En ce qui concerne la méthode d'évaluation intrinsèque, nous avons introduit la notion de la stabilité de polarités *sentiment stability (SS)* dans les espaces d'embeddings. Pour l'évaluation extrinsèque, nous avons évalué les performances du réseau convolutif CNN, dans ses versions statique et non statique, avec les embeddings spécifiques à l'arabe.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion

Cette thèse se situe dans le cadre de l'analyse d'opinions en arabe. L'analyse d'opinions (AO) consiste à identifier la subjectivité et la polarité (positive, négative, neutre) d'un énoncé donné. Elle s'applique au niveau du document, de la phrase ou d'un groupe de mots. L'objectif de cette thèse consiste à prédire la polarité d'un document.

Dans cette thèse, nous nous focalisons sur la détection de polarité par des méthodes à base de réseaux de neurones artificiels pour la langue arabe. Dans ce domaine, la majorité des réseaux neuronaux utilisent comme entrée des représentations vectorielles continues (*embeddings*) de mots.

Ce manuscrit met en avant les embeddings appliqués à la tâche d'analyse d'opinions. L'utilisation de ces embeddings est justifiée par leurs performances prometteuses pour plusieurs tâches du traitement automatique des langues naturelles, entre autre l'analyse d'opinions.

Ce travail de thèse porte sur plusieurs axes. Dans un premier temps, une étude préliminaire a été menée sur l'utilisation des différentes ressources d'embeddings de mots pré-entrainés existantes. L'architecture CNN de [Dahou *et al.*, 2016] a été téstée dans 3 cadres de classification : binaire, ternaire et quinaire. Nous avons tout d'abord réglé quelques hyperparamètres du CNN en proposant un protocole spécifique à l'analyse d'opinions permettant de choisir la longueur du document pour le CNN et le type de padding/truncating. La qualité des espaces d'embeddings de mots pré-entrainés existants vis à vis de la tâche d'analyse d'opinions a été étudiée en introduisant la notion de ratios de positivité et de négativité. La couverture du corpus par les espaces d'embeddings impacte également les performances du CNN.

Les embeddings de mots pré-entrainés existants représentent des vecteurs à valeurs continues au niveau des mots. Or, un mot arabe, défini comme une séquence de caractères délimitée par deux séparateurs, est complexe étant donné l'agglutination et la richesse morphologique de l'arabe. Une phrase peut être composée d'un seul mot. Dans cette perspective, nous avons supposé qu'une décomposition en éléments simples du mot complexe pourrait améliorer la qualité des embeddings. C'est pourquoi,

dans un deuxième temps, nous avons proposé des embeddings spécifiques à l'arabe prenant en compte les spécificités de cette langue. L'idée consiste à explorer les embeddings spécifiques à la langue arabe afin de trouver la meilleure unité lexicale pour l'analyse d'opinions.
Nous avons défini les différentes représentations possibles d'un mot obtenant ainsi 6 unités lexicales : mot, token, token\clitiques, lemme, light stemme et stemme. Nous avons entrainé, pour chaque unité lexicale, word2vec et FastText pour obtenir à la fin 12 espaces d'embeddings.
Pour choisir l'unité lexicale la plus adéquate pour l'AOA, nous avons testé deux architectures neuronales de nature différente : un réseau convolutif (CNN) et un réseau récurrent de type BiLSTM. Les meilleures performances du CNN et BiLSTM sont obtenues au niveau lemme. L'unité lexicale *lemme* semble ainsi être l'unité la plus adéquate pour l'AOA. Un gain de 1.2% en exactitude est obtenu avec les embeddings de lemmes.
Par ailleurs, des combinaisons des sorties de systèmes neuronaux et de leurs entrées ont été réalisées. Les résultats obtenus avec les différentes combinaisons ne montrent pas d'amélioration.

Dans un troisième temps, nous avons étudié la qualité des embeddings spécifiques à l'arabe pour la tâche d'AO. Deux méthodes d'évaluation d'embeddings ont été testées : une méthode intrinsèque et une autre extrinsèque. En ce qui concerne la méthode d'évaluation intrinsèque, nous avons introduit la notion de la stabilité de polarités *sentiment stability (SS)* dans les espaces d'embeddings. Pour l'évaluation extrinsèque, nous avons évalué les performances du réseau convolutif CNN, dans sa version statique et non statique, avec les embeddings spécifiques à l'arabe.

Le tableau 6.1 résume l'impact des expériences phares réalisées dans le cadre de classification binaire (positive, négative) ainsi que leurs gains. Le travail de [Dahou *et al.*, 2016] représente la baseline dans cette thèse donnant 89.6% d'exactitude.

Configuration	A	Unité lexicale	Embeddings		Version CNN	Taille document	Padding / truncating	Couverture
			Type	Nom				
Baseline	89.6	mot	w2v	Dahou	non statique	882	pré	67
config. 1	89.6	mot	w2v	wiki-cbow	non statique	300	post	67
config. 2	90.3	mot	FT	cc.arz	non statique	300	post	100
config. 3	91.5	lemme	w2v	ArEmb	non statique	300	post	100
config. 4	91.9	lemme	w2v	ArEmb	statique	300	post	100

TABLE 6.1 – Récapitulation de l'impact des expériences phares réalisées. Les performances sont représentées par l'exactitude (A)

L'architecture neuronale utilisée dans [Dahou *et al.*, 2016] est un réseau convolutif dans sa version non statique. Les hyperparamètres relatifs à la longueur de documents et du type de padding sont fixés au hasard. Nous avons défini un protocole pour déterminer la longueur et le type de padding/truncating. Le CNN avec les nouveaux valeurs d'yperparamètres a été testé avec les différents ensembles d'embeddings de mots pré-entraînés existants (de type word2vec et FastText). La performance obtenue

avec les embeddings de mots de type word2vec *wiki-cbow* permettant de couvrir 67% du corpus, est égale à celle obtenue avec la baseline (89.6%) : c'est la *config. 1*. La performance obtenue avec les embeddings de mots de type FastText *cc.arz* couvrant 100% le corpus est 90.3% (c'est la *config. 2*. Un gain de 1.2 est obtenu, dans la *config .3*, avec les embeddings spécifiques à l'arabe et plus précisément avec les embeddings de lemmes Dans toutes les configurations *config. 1*, *config. 2*, *config. 3*, le réseau CNN a été utilisé dans sa version non statique, c'est à dire que les embeddings d'entrée sont mis à jour lors de l'entraînement du CNN. L'utilisation de la version statique du CNN dans la *config. 4* avec les embeddings de lemmes améliore les performances

Les embeddings spécifiques à la langue arabe constituent la contribution fondamentale de cette thèse. Les résultats sont prometteurs. Nous les mettons disponibles à la communauté scientifique. Ils sont téléchargeables gratuitement via le lien suivant : https://lium.univ-lemans.fr/arsentimentanalysis/.

6.2 Perspectives

Les contributions exposées dans cette thèse nous ont amené à envisager de nouvelles pistes de recherche ou à approfondir davantage celles ayant déjà été étudiées. Cette section vise à décrire ces perspectives afin de poursuivre les recherches entamées dans cette thèse.

En ce qui concerne les embeddings spécifiques à la langue arabe, nous proposons, tout d'abord, d'approfondir l'étude qualitative sur l'évaluation extrinséque des embeddings spécifiques à l'arabe. Il est intéressant, d'une part, de déterminer, s'il existe, des rapprochements sémantiques intéressants après l'apprentissage des embeddings spécifiques à la tâche. D'autre part, il est important d'étudier l'explicabilité et l'interprétabilité du réseau convolutif (CNN). L'explicabilité permet de prendre compte explicitement de la classe de prédiction à partir des données et leurs caractéristiques. Autrement dit, étudier la possibilité de mettre en relation les valeurs prises par certaines caractéristiques et leurs conséquences sur la prédiction. Quant à l'interprétabilité, elle consiste à étudier la possibilité d'identifier les caractéristiques pertinentes qui participent le plus à la prédiction, voire même de quantifier leurs importances.

Une difficulté de la prédiction de la classe négative a été observée dans le cadre de la classification binaire. La première explication met l'accent sur le désequilibre entre les classes positive et négative dans le corpus d'apprentissage. La proportion des données de la classe négative resprésente uniquement 16% du corpus d'apprentissage. Pour résoudre ce probléme, des techniques de réechantillonnage peuvent être appliquées afin d'équilibrer les proportions de classes dans le corpus. Le sous-échantillonnage de la classe majoritaire et le sur-échantillonnage de la classe minoritaire, deux techniques basique de réechantillonnage, ont été testés avec les embeddings de mots pré-entraînés existants et n'ont pas amélioré les performances. Il serait intéressant de tester d'autes techniques plus sophistiquées ciblant le problème de désequilibre des classes dans le corpus d'apprentissage telles que :

— la méthode SMOTE (*Synthetic Minority Oversampling Technique*) qui augmente le nombre de données par interpolation des voisins [Chawla *et al.*, 2002]
— le seuillage (*thresholding*) dans le réseau CNN qui ajuste le seuil de la décision en changeant les probabilités des classes [Lawrence *et al.*, 1998]
— La méthode d'appprentissage sensible aux coûts (*cost sensitive learning*) qui attribue différents coûts aux exemples malclassés [Elkan, 2001]
— combinaison des méthodes citées ci-dessus

Les embeddings spécifiques à la langue arabe ont été testé dans le cadre de la classification binaire et ils ont permis d'améliorer les performances du systèmes d'analyse d'opinions. Les cadres de classification ternaires et quinaire ont été évalués avec les embeddings de mots pre-entrainés existants et ont montré une difficulté de prédiction de la classe neutre et une incertitude des classes positive et très positive due à la limite floue entre ces deux classes (il en est de même pour les classes négatives et très négative). Il serait préférable d'évaluer les cadres de classification ternaire et quinaire avec nos embeddings spécifiques à l'arabe afin de déterminer si ces embeddings améliorent les performances de la classification multi-classes.

Les meilleures performances dans cette thèse sont obtenues avec les embeddings de lemmes de type word2vec. Nous avons menée une analyse qualitative des embeddings de type word2vec. En plus des embeddings lemmes, nous avons analysé également les embeddings de mots afin d'étudier les possibles différences entre ces deux unités lexicales. Il est préférable de mener la même analyse qualitative sur embeddings de type FastText.

Une autre perspective réside dans l'intégration de connaissances hétérogènes extérieures aux réseaux neuronaux. Ceci revient à concevoir une approche hybride pour l'analyse d'opinions en arabe. Pour ce faire, nous pouvons intégrer notre lexique *ArSentLex* lors de l'apprentissage du CNN. Nous pouvons intégrer également un lexique d'émoticônes qui semblent être, en plus de mots polarisés, porteurs fondamentaux de polarité et de sentiments.

Le problème d'analyse d'opinions est réduit, dans cette thèse, à la détéction de la polarité. Il comprend d'autres tâches telles que la détermination de l'intensité de la polarité, l'identification de l'entité sur laquelle porte l'opinion et ses différents aspects, déterminantion de la polarité par aspect, étude de l'évolution d'opinions sur une entité donnée en fonction du temps. Il est intéressant d'élargir notre champs de recherche en analyse d'opinions et considérer,en plus de la détection de la polarité, autres tâches d'AO.

Une autre réflexion consiste à tester nos embeddings spécifiques à la langue arabe pour d'autre tâches de traitement automatique de langues naturelles, telles que le résumé automatique et la recherche d'information.

Annexe A

Embeddings spécifiques à la langue arabe : statistiques et performances

A.1 Prototocole de représentation de documents : impact sur les performances

Le tableau A.1 rapporte les performances du CNN sur le corpus de validation *Dev*. Les lignes grisées reprèsentent les meilleures performances obtenues.

	Emb	***CNN_300-post***				***CNN_DAHOU***		**Impact**
		A	**P**	**R**	**F1**	**A**	**inter_conf**	
2 classes	wiki-cbow	89.4	83.94	75.14	79.30	89.3	[88.3, 90.2]	+0.1
	cc.arz	**90.1**	86.70	75.41	80.66	90.2	[89.2, 91.1]	-0.1
3 classes	Dahou	72.5	59.03	51.82	55.19	72.7	[71.4, 73.9]	-0.2
	cc.ar	**73.3**	60.42	53.78	56.91	73.5	[72.2, 74.7]	-0.2
5 classes	twt-cbow	**55.6**	51.00	47.20	49.03	55.5	[54.1, 56.8]	+0.1
	cc.arz	55.4	52.14	45.87	48.81	55.7	[54.3, 57.0]	-0.3

TABLE A.1 – Performances du CNN sur le corpus de Dev avec longueur 300.

Les performances du *CNN_300-post* sont proches de celles de *CNN_DAHOU*. Au niveau exactitude, on note des gains et des pertes selon les embeddings pré-entrainés utilisés. Néanmoins, les exactitudes de *CNN_300-post* sont incluses dans les intervalles de confiance des performances de *CNN_DAHOU*. On pourrait ainsi juger les gains et les pertes comme négligeables et non significatifs.

A.2 Résulats des Embeddings combinés

Les tableaux A.2 et A.3 rapportent les performances du CNN sur le corpus de validation *Dev* avec les embeddings combinés aux niveaux des unités lexicales *lemme* et *mot* repectivement. La

partie grisée représente les caractéristiques de la combinaison donnant les meilleures performances sur le corpus de Dev.

Embeddings	Dimension	A	P	R	F1
EmbConcat	600	91.1	84.18	78.67	81.33
EmbACP	400	**91.4**	85.17	78.8	**81.86**
	300	**91.4**	86.16	77.85	81.80
	200	91.3	85.88	77.38	81.41
	100	90.7	84.90	75.55	79.96
EmbAutoEnc	400	90.68	84.24	76.12	79.97
	300	90.9	86.49	74.60	80.11
	200	90.84	85.52	75.34	80.10
	100	90.44	83.18	76.36	79.63

TABLE A.2 – Évaluation des différents protocoles de combinaison d'embeddings de lemmes sur le Dev.

En ce qui concerne l'unité *lemme*, les résultats de combinaison sont reportés dans le tableau A.2. La meilleure performance (91.4%) est obtenue avec les embeddings de lemmes (de dimension 400) obtenus avec la combinaison *ACP* d'embeddings de lemmes entraînés avec w2v et FT.

Embeddings	Dimension	A	P	R	F1
EmbConcat	600	**91.2**	88.43	74.29	80.74
EmbACP	400	**91.2**	85.75	76.92	**81.09**
	300	**91.2**	86.02	76.59	81.03
	200	90.9	85.95	75.46	80.36
	100	90.5	85.01	74.27	79.28
EmbAutoEnc	400	89.84	83.17	72.83	77.65
	300	90.02	84.27	72.49	77.94
	200	90.16	84.26	73.23	78.36
	100	89.96	83.14	73.56	78.05

TABLE A.3 – Évaluation des différents protocoles de combinaison d'embeddings de mots sur le Dev.

En ce qui concerne l'unité *mot*, les résultats de combinaison sont reportés dans le tableau A.3. La meilleure performance (91.2%) est obtenue avec les embeddings de mots (de dimension 400) obtenus avec la combinaison *ACP* d'embeddings de mots entraînés avec w2v et FT. Nous comparons maintenant les résultats de combinaison aux niveaux lemme et mot. Quelque soit l'approche de combinaison, plus la dimension d'embeddings est petite, plus les performances se dégradent.

Annexe B

La traduction automatique pour l'AOA : une étude empirique

Sommaire

B.1 Motivations 140
B.2 État de l'art 141
B.3 Méthodologie 142
B.3.1 Systèmes d'analyse d'opinions 144
B.3.2 Embeddings de documents 144
B.4 Résultats et discussion 145
B.4.1 Performances sur le corpus *LABR_ar* 145
B.4.2 Performance sur le corpus *LABR_en* 147
B.4.3 Discussion 148
B.5 Conclusion 150

Étant conscients de la complexité de la langue arabe et du nombre réduit de ressources disponibles pour l'analyse d'opinions en arabe, nous avons recours à la traduction automatique de l'arabe vers l'anglais. Avec le progrés considérable dans les systèmes de traduction automatique, nous supposons que la complexité spécifique à la langue arabe peut être réduite, voire disparaître, en traduisant le texte arabe vers l'anglais (une langue non agglutinante et non riche morphologiquement). Nous mesurons l'impact de la traduction automatique sur l'analyse d'opinions en arabe.

Nous nous intéressons à mesurer l'impact de la traduction automatique pour la tâche d'analyse d'opinions en arabe. Nous utilisons l'anglais comme langue cible de la traduction automatique. Nous choisissons d'appliquer le système de [Le & Mikolov, 2014] donnant de bonnes performances pour l'Analyse d'opinions en anglais.

Nous décrivons les motivations dans la section B.1. Nous présentons l'état de l'art des travaux en AOA utilisant la traduction automatique B.2. Nous détaillons ensuite notre méthodologie dans la

section B.3. Puis, nous présentons et discutons les résulats dans la section B.4. En fin, nous concluons dans la section B.5.

B.1 Motivations

Une plus grande quantité de travaux en analyse d'opinions a été réalisée pour la langue anglaise, et relativement peu de travaux en arabe. Pour traiter l'AO en arabe, la traduction automatique des ressources en anglais ou des textes arabes peut être appliquée pour construire des systèmes d'AOA. Les ressources développées en analyse d'opinions en arabe ne sont pas nombreuses et parfois indisponibles [Al-Kabi *et al.*, 2016; Boudad *et al.*, 2017]. Dans l'analyse d'opinions, la construction de ressources fiables est coûteuse et nécessite du temps car elle a besoin d'experts pour annoter les corpus et construire des lexiques polarisés ou ontologies d'opinions (*sentiment ontology*). Pour éviter le coût de la mise en place de telles ressources, une piste qui peut être appliquée est de traduire les ressources existantes d'une langue peu dotée en ressources linguisiques (*low-resources language*) vers une langue bien dotée et d'appliquer des méthodes efficaces pour l'AO dans cette langue.

Par ailleurs, les progrès de la traduction automatique (*Machine Translation* MT) ont fortement bouleversé les systèmes de traitement automatique des langues (TAL). La traduction automatique est intégrée dans de nombreuses applications TAL telles que les services de traduction en ligne, la recherche et l'extraction d'informations, *etc.* La recherche en analyse d'opinions a également tiré profit de la traduction automatique, en particulier pour les langues avec peu de ressources [Mohammad *et al.*, 2016; Can *et al.*, 2018].

Dans les chapitres précédents (4 et 5), nous nous sommes intéressés à la spécificité de la langue arabe, et nous avons proposé de nouveaux embeddings prenant en compte les caractéristiques de l'arabe. Cette partie représente une autre piste de recherche totalement différente. Son idée se base d'un côté sur la difficulté de l'arabe et le manque relatif de ressources disponibles en AOA et, d'un autre côté sur le progrés réalisé dans le domaine de la traduction automatique.

Les systèmes actuels de traduction automatique de l'anglais vers l'arabe et inversement sont performants. Nous choisissons dans ce travail de traduire de l'arabe vers l'anglais et entraîner un système d'AO sur l'anglais. Nous supposons que la traduction automatique de textes arabes en anglais va résoudre les spécificités de la langue arabe, principalement l'agglutination et la richesse morphologique. Autrement dit, la prise en compte de la spécificité de l'arabe est faite au niveau du module de la traduction automatique. Dans cet chapitre, nous étudions l'impact de la traduction automatique de l'arabe vers l'anglais pour la tâche d'AOA. Nous répondons à plusieurs questions de recherche telles que :

1. Les systèmes de traduction automatique modifient-ils la subjectivité ou la polarité initiales exprimées dans le texte arabe source ?

2. Quelles performances sont obtenues par les systèmes d'AO utilisant la traduction automatique ? Sont-elles meilleures que celles obtenues avec un système d'analyse de textes en arabe ?
3. Quel intérêt y a-t-il d'utiliser la traduction auomatique dans l'AOA ?
4. L'utilisation de plus de données d'apprentissage améliore-t-elle les performances même si des données supplémentaires proviennent d'un autre domaine ?

B.2 État de l'art

La recherche en AOA se base sur des ressources de type corpus et lexiques polarisés. Nous avons résumé, dans la section 2.3.3 du chapitre 2, les différentes ressources existantes pour l'AOA. Le nombre de ces ressources est relativement petit par rapport à celui des ressources disponibles en anglais. Certains travaux en AOA ont recours à la traduction automatique pour créer des ressources en arabe et en anglais. Les ressources traduites peuvent être considérées comme fiable vu les bonnes performances des systèmes de traduction automatique.

Dans cette section, nous présentons quelques travaux qui ont utilisé la traduction automatique pour construire des systèmes d'AOA. Dans ce cadre, deux pistes sont possibles : soit traduire de l'arabe vers l'anglais et entraîner un système sur les textes anglais, soit traduire les ressources de l'anglais vers arabe et entraîner un système sur les textes arabes [Salameh *et al.*, 2015; Mohammad *et al.*, 2016]. La deuxième piste a été déjà testée pour l'AOA. Dans ce travail, nous procédons selon la première piste.

Pour s'assurer de la fiabilité de textes traduits automatiquement par un système de MT dans le cadre de l'AO, [Mohammad *et al.*, 2016] a comparé les performances du système d'AO sur les textes traduits automatiquement et ceux traduits manuellement avec des experts. Il a conclu qu'il n'y avait pas de différence au niveau performances des systèmes d'AO. Il est donc possible d'utiliser des ressources traduites pour améliorer les performances des systèmes d'AOA et renforcer ainsi leurs capacité de généralisation [Elnagar *et al.*, 2017].

[Al-Shabi *et al.*, 2017] explore la classification d'opinions multilingues de l'anglais vers l'arabe, sans aucun effort d'annotation manuellelle. Il a constaté qu'il est facile à construire un système d'analyse performant sans avoir besoin d'une analyse linguistique approfondie. Il a conclu qu'un bon modèle de classification peut être obtenu à partir de corpus traduits malgré le bruit ajouté par la traduction automatique.

Il est bien connu que la construction d'un système d'AO nécessite des corpus et/ou des lexiques. Certaines recherches traduisent des corpus et d'autres traduisent des lexiques polarisés [Refaee & Rieser, 2015; Mohammad *et al.*, 2016].

La traduction automatique a été utilisée principalement pour résoudre le problème de manque de ressources d'AO disponibles en arabe, principalement des lexiques polarisés. En fait, plusieurs travaux

ont été réalisées en traduisant le lexique anglais *SentiWordNet* [1]. Plusieurs lexiques résulatants de la traduction de *SentiWordNet* :

— le lexique arabe *SLSA* disponible gratuitement [Eskander & Rambow, 2015]

— les lexiques non disponibles de [Duwairi *et al.*, 2015; Refaee & Rieser, 2015; Alotaibi & Anderson, 2016]

De plus, Guellil *et al.* [2018b] a exploité la traduction automatique pour construire un lexique polarisé pour le dialecte algérien (contenant des mots en arabe et d'autres dialectaux).

Par ailleurs, la construction par extension automatique représente une méthode de construction de lexiques polarisés. Elle se base sur une liste de mots polarisés fixée manuellement, étendue par la suite par application de la PMI sur un corpus (comme détaillé dans la section 2.3.1 du chapitre 2). La traduction automatique peut être utilisée pour obtenir la liste initiale de mots polarisés traduits en arabe et l'étendre ensuite par application de la PMI sur un corpus arabe [Mahyoub *et al.*, 2014]. Pour agrandir le lexique, deux façons ont été explorées :

— traduire les synonymes [Ibrahim *et al.*, 2015a]

— concaténer des lexiques polarisés en arabe et en anglais pour classer des textes multilingues (contenant des mots en anglais et en arabe) [Al-Horaibi & Khan, 2016]

La majorité des travaux conclut que la traduction apporte des résultats compétitifs. Cependant, El-Beltagy [2017] a montré que l'exactitude du système d'AO baisse avec l'utilisation d'un lexique traduit et que la qualité de ce lexique n'est pas aussi élevée qu'un lexique construit manuellement. Il est ainsi important que le système de traduction automatique soit performant.

B.3 Méthodologie

Dans ce travail, nous nous intéressons à l'étude de l'impact de la traduction automatique sur l'analyse d'opinions en arabe. La majorité des travaux existants traduisent de l'anglais vers l'arabe les ressources disponibles en analyse d'opinions en anglais [Mohammad *et al.*, 2016; Al-Shabi *et al.*, 2017; Elnagar *et al.*, 2017]. Nous procédons différemment dans ce travail. Nous traduisons le corpus de l'arabe vers l'anglais et nous entraînons les mêmes systèmes d'AO sur le corpus en arabe et le corpus traduit en anglais.

Nous considérons le système d'AO entraîné sur le corpus arabe comme une *baseline* et nous le comparons avec le système entraîné le corpus traduit en anglais. Nous pouvons ainsi mesurer l'impact de la traduction automatique du texte arabe en anglais, c'est-à-dire voir si la traduction automatique a modifié ou pas la polarité des textes.

La mesure de l'impact de la traduction automatique sur l'alalyse d'opinions en arabe est illustée dans la figure B.1. Nous suivons dans l'ordre les étapes du schéma expérimental ci-dessous :

1. Identifier le corpus en arabe *corpus_ar*

1. http ://sentiwordnet.isti.cnr.it/

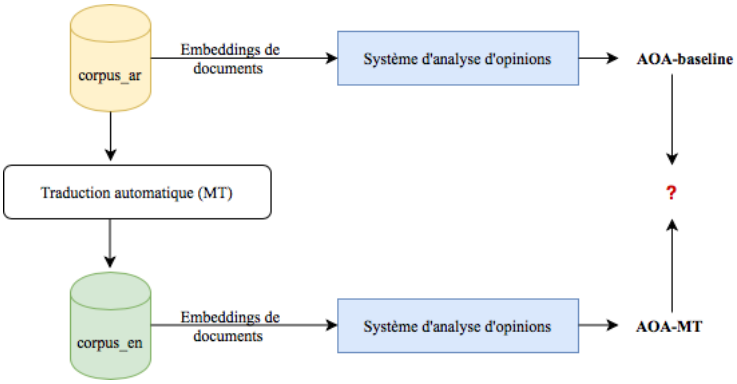


FIGURE B.1 – Méthodologie pour mesurer l'impact de la traduction automatique sur l'AOA.

2. Entraâner un système d'AO sur l'ensemble d'apprentissage du *corpus_ar* basé sur les embeddings de documents arabes pré-entraînés. Ce système est désigné par *AOA-baseline*
3. Tester le système AOA-baseline sur l'ensemble de test du *corpus_ar* et calculer ses performances.
4. Traduire *corpus_ar* de l'arabe vers l'anglais et obtenir la version anglaise *corpus_en*
5. Entraîner un système d'AO sur l'ensemble d'apprentissage du *corpus_en* basé sur des embeddings de documents anglais pré-entraînés. Ce système est noté *AOA-MT*
6. Tester le système *AOA-MT* sur l'ensemble de test du corpus *corpus_en* et calculer ses performances
7. Comparer les performances des deux systèmes *AOA-baseline* sur *corpus_ar* et *AOA-MT* sur *corpus_en* et déduire des conclusions.

Afin de traduire des textes arabes en anglais, nous utilisons un des systèmes de traduction automatique développé au sein du laboratoire LIUM [2]. Il est basé sur le moteur de traduction automatique statistique *Moses* [3] [Koehn *et al.*, 2007]. Il prend de grandes quantités de données parallèles (arabe/anglais) et utilise des co-occurrences de mots et de phrases pour déduire des correspondances de traduction entre les deux langues. Pour le décodage, *Moses* trouve la phrase ayant le score le plus élevé dans la langue cible (ici, l'anglais).

Nous utilisons le corpus LABR [Nabil *et al.*, 2014] pour entraîner les systèmes d'AO. La version de LABR en arabe est nommée *LABR_ar*. La version de LABR en anglais est nommée *LABR_en* et est obtenue par traduction de *LABR_ar* avec notre système de traduction automatique.

Nous présentons, dans la section B.3.1, les systèmes d'AO utilisés. Ces derniers prennent comme entrée des embeddings de documents présentés dans la section B.3.2.

2. https://lium.univ-lemans.fr/
3. http://www.statmt.org/moses/

B.3.1 Systèmes d'analyse d'opinions

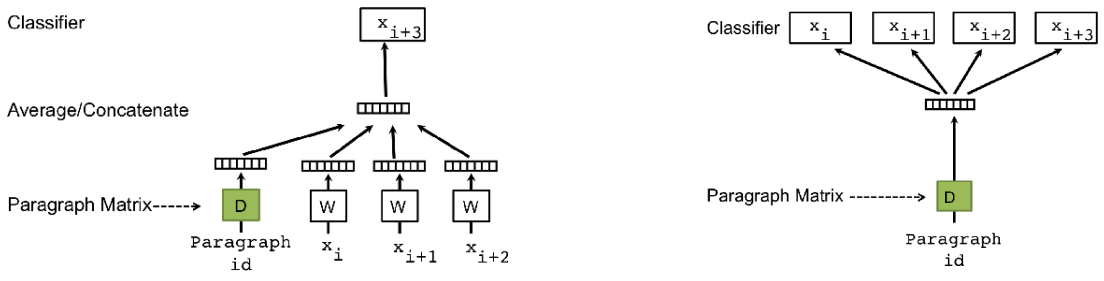
Notre idée consiste à reproduire le système de [Le & Mikolov, 2014] donnant de bonnes performances pour l'Analyse d'opinions en anglais et le tester pour l'AOA. Nous utilisons deux classificateurs : une régression logistique (RL) et un perceptron multicouche (MLP). Dans ce travail, nous nous situons dans le cadre de la classification binaire. Chacun des systèmes prédit, pour un document donné, la polarité *positive* ou *négative*.
La régression logistique est présentée dans la section 1.2.2.1 du chapitre 1. Le perceptron multicouches est présenté dans la section 3.1.1.2 du chapitre 3. Le MLP contient 3 couches : la couche d'entrée dont le nombre de neurones est égal à la taille du vecteur d'entrée au classifieur, une couche cachée avec 50 neurones et la couche de sortie avec 2 neurones pour prédire la polarité du texte d'entrée : positive ou négative.
Nous avons entraîné les classificateurs avec les différents taux d'apprentissage 10^{-3}, 10^{-4} et 10^{-5}. Dans la suite, nous reportons les meilleurs résultats obtenus avec le taux d'apprentissage 10^{-5}.

Les classifieurs prennent commen entrée des embeddings de documents présentés dans la section B.3.2.

B.3.2 Embeddings de documents

Les embeddings de documents ont été utilisés pour l'analyse d'opinions en anglais par [Le & Mikolov, 2014] obtenant les meilleures performances avec un embedding de document comparé à d'autres approches sur le corpus anglais IMDB [Maas *et al.*, 2011] qui contient 100 000 critiques sur films. Motivés par leur travail, nous proposons d'utiliser les embeddings de documents pour l'AOA. Nous voulons mesurer l'efficacité de la méthode de [Le & Mikolov, 2014] pour l'analyse d'opinions en arabe.

L'algorithme *paragraph vector* permet d'obtenir des représentations distribuées (Doc2vec) pour n'importe quelle séquence de longueur variable, allant des phrases aux documents. Il calcule les représentations vectorielles des documents dans un espace vectoriel multi-dimensionnel. Les vecteurs de mots sont situés dans l'espace vectoriel où les mots ayant des similarités sémantiques sont proches dans l'espace vectoriel.



(A) Version DM (B) Version DBOW

FIGURE B.2 – Les versions DM et DBOW de l'algorithme *paragraph vector* [Le & Mikolov, 2014]

L'algorithme *paragraph vector* dispose de deux façons de construction d'embeddings de documents. La première, dite *distrubuted memory* (DM), consiste à prédire un mot sachant ses mots précédents et l'embedding du texte. La deuxième, dite *distributed bag of words* (DBOW), consiste à prédire un groupe aléatoire de mots dans le texte étant donné l'embedding du texte. La figure B.2 illustre les versions DM et DBOW.

L'embedding du document est une concaténation des deux vecteurs entraînés, l'un avec la version à mémoire distribuée *DM* et l'autre avec la version distribuée de mots *DBOW*, chacun ayant 400 dimensions. Ainsi, 800 est la dimension du vecteur d'entrée du classificateur.
Nous avons conservé les mêmes hyperparamètres de l'algorithme *paragraph vector* utilisés par [Le & Mikolov, 2014].

B.4 Résultats et discussion

Nous présentons, dans un premier temps, les performances des systèmes d'AO sur le corpus arabe *LABR_ar* (section B.4.1), et dans un deuxième temps, les performances des systèmes d'AO sur le corpus anglais *LABR_en* (dans le section B.4.2). Nous comparons et discutons les résultats dans la section B.4.3.

B.4.1 Performances sur le corpus *LABR_ar*

Dans cette section, nous testons deux classifieurs : MLP et RL sur le corpus *LABR_ar*.
Nous avons testé différents prétraitements du corpus *LABR_ar* :

- sans aucun prétraitement noté ∅
- le prétraitement *caractères spéciaux* qui consiste à séparer les ponctuations des mots et à les considérer comme des mots normaux. Certains caractères spéciaux tels que ! ? sont porteurs de polarité. Certaines combinaisons de ces caractères spéciaux, par exemple :) :(, sconstituent des émotocônes qui sont significatifs pour la tâche d'AO. Il est donc important de les considérer comme des mots.
- le prétraitement *light stemming* qui consiste à appliquer l'outil de light stemming [4] sur le corpus *LABR_ar*

Les performances des deux classifieurs obtenus dans le cadre de la classification binaire sont rapportées dans le tableau B.1. La meilleure performance (76.7%) est obtenue avec la régression logistique sur le corpus *LABR_ar* après light stemming. Pour déterminer le meilleur prétraitement à l'AO, nous comparons les performances obtenues avec les différents prétraitement avec les deux classifieurs. Nous observons que pour la régression logistique, la meilleure performance est obtenue avec le prétraitement de type light stemming. Nous déduisons la même remarque pour le perceptron multicouches. D'où, l'utilité du light stemming pour la tâche d'AOA.

4. https ://github.com/motazsaad/arabic-light-stemming-py

	Régression logistique	Perceptron multicouches
∅	74.4%	75.3%
caractères spéciaux	74.6%	74.5%
Light stemming	**76.7%**	76.6%

TABLE B.1 – Exactitude des classifieurs RL et MLP sur *LABR_ar* dans le cadre d'une classification binaire

Les embeddings de documents sont calculés en se basant sur les embeddings de mots de type word2vec. Ils peuvent donc capturer la similarité sémantique entre les mots. Par exemple, les mots جيد "bon" et ممتاز "excellent" sont proches l'un de l'autre. Pour mesurer l'efficacité de l'algorithme *paragraph vector* pour la langue arabe, nous recherchons les 10 premiers mots similaires au mot جيد (bon) qui sont dans l'ordre suivant : جميل (beau), رائع (fabuleux), ممتع (agréable), مفيد (utile), شيق (intéressant), ممل (ennuyeux), خفيف (léger), ممتاز (excellent), لطف (sympatique), جدا (très). Parmi ces mots, sept mots sont sémantiquement similaires à جيد (bon). Nous remarquons quelques erreurs de similarité :

— Le mot ممل (ennuyeux) est proche de جيد (bon), ce qui n'est pas vrai.

— Le mot ممل (ennuyeux) est plus proche de جيد (bon) que ممتاز (excellent), ce qui est faux.

Les erreurs dans les top n voisins les plus proches sont fortement liée à la taille du corpus d'apprentissage utilisé dans l'apprentissage de l'algorithme *paragraph vector*. Puisque *paragraph vector* est un algorithme d'apprentissage non supervisé, nous pourrions améliorer la qualité des embeddings en ajoutant plus d'exemples au corpus d'apprentissage.

En plus de la classification binaire, nous avons également testé les classifieurs RL et MLP pour la classification multi-classes.

	Régression logistique	**Perceptron multicouches**
Exactitude	32.4%	**30.6%**

TABLE B.2 – Exactitude des classifieurs RL et MLP sur *LABR_ar* dans le cadre d'une classification multi-classes

Les performances obtenues dans le cadre de classification multi-classes sont reportées dans le tableau B.2. Dans ce cadre, nous avons choisi le prétraitement *light stemming* donnant les meilleurs résultats dans le cadre de la classification binaire. Le tableau B.2 montre que la régression logistique est plus efficace que MLP. En effet, l'exactitude de la régression logistique est supérieure à celle du MLP (32.4% vs. 30.6%).

De plus, les performances dans le cadre de classification binaire sont supérieures à ceux du cadre

multi-classe. En effet, avoir plus de classes n'est pas le seul défi imposé par la classification multi-classes. L'autre difficulté vient de la relation entre certaines classes, c'est-à-dire la relation entre les polarités positives et très positives [5]. Dans ce travail, nous avons adopté une classification plate (tableau B.3) et nous avons obtenu une exactitude égale à 32.4% en utilisant un classificateur de régression avec des embeddings de documents. Cependant, [Shboul *et al.*, 2015] a utilisé muti Naive Bayes avec les vecteurs de document obtenu avec la méthode BOW. Ils ont obtenu 45% d'exactitude. Tous les travaux mentionnés dans le tableau B.3 sont effectués sur LABRar.

	Exactitude	**Hiérarchie**	**# niveaux**
Notre système RL	32.4%	plate	1
[Shboul *et al.*, 2015]	45%		
[Al-Ayyoub *et al.*, 2016]	45.7%		
	46.2%	multi-niveaux	2
	57.8%		4

TABLE B.3 – Comparaison des travaux effectués sur le corpus LABR dans le cadre classification multi-classes

Al-Ayyoub *et al.* [2016] (tableau B.3) prouve que la hiérarchie multi-niveaux améliore les performances de la classification multi-classes. Ils ont utilisé le classifieur KNN et ont obtenu une exactitude égale à 46,2% avec une hiérarchie à 2 niveaux. Et ils ont obtenu une précision de 57,8% avec une hiérarchie à 4 niveaux.

B.4.2 Performance sur le corpus *LABR_en*

La régression logistique est plus performante que le perceptron multicouches pour la classification du corpus *LABR_ar*. Le tableau B.4 rapporte les performances de la régression logistique sur les corpus *LABR_ar* et *LABR_en* dans le cadre de la classification binaire.

Système	**Corpus**	**Exactitude**
AOA-baseline	LABR_ar	74.6%
AOA-MT	LABR_en	76.3%

TABLE B.4 – Performances de la régression logistique sur les corpus *LABR_ar* et *LABR_en*

L'exactitude des systèmes *AOA-baseline* et *AOA-MT* sont respectivement 74.6% et 76.3%. Nous notons un gain de 2 points avec le système *AOA-MT*. Ce dernier est entraîné sur le corpus *LABR_en*. Nous rappellons que *LABR_en* représente la traduction automatique en anglais du *LABR_ar* utilisé dans le système AOA-baseline. Nous déduisons que la traduction automatique ne semble pas modifier la polarité et apporte une performance meilleure à celle obtenue avec le corpus arabe.

5. la même relation existe entre les polarités négatives et très négatives

B.4.3 Discussion

Afin d'expliquer et de comprendre pourquoi la traduction de l'ensemble de données LABR arabe en anglais améliore les performances, nous avons analysé le corpus traduit LABR_en. Une transformation effectuée au cours du processus de traduction consiste à supprimer tous les mots non traduits de LABR_en car ces mots peuvent être ambigus ou bruyants dans l'ensemble de données anglais. Compte tenu de ce choix et des résultats observés, nous avons fait l'hypothèse que les mots non traduits peuvent être ambigus. Pour valider cette hypothèse, nous avons effectué une analyse de mot arabe non traduit. Nous remarquons que ces mots non traduits sont principalement : mots dialectaux, noms propres, mots typographiques, mots avec répétition de caractères, mots d'origine non arabe écrits avec des lettres arabes (tels que : الرفيو / Alrfyw / for *review*, البروتكشن / Albrwtk / for *protection*, *etc.*).

Le tableau B.5 donne des exemples dans *LABR_ar* et leurs traductions en anglais dans *LABR_en*.

Texte en arabe	Texte en anglais
مش وحش خفيف	it 's not bad خفيف
مستمتعة بيه جدّا	it 's enjoying sir جدّا
راااايع جدّاا. بسيط و مفهوم	راااايع very much. is simple and understandable

TABLE B.5 – Exemples en arabe dans corpus *LABR_ar* et leurs équivalents traduits en anglais dans *LABR_en*

Suite à cette analyse, nous considérons que ces mots non traduits semblent perturber la détection de la polarité, nous choisissons donc de considérer ces mots comme bruit. Afin de voir si les informations nécessaires sont conservées dans le corpus anglais pour détérminer la polarité, nous choisissons de supprimer les mots bruit du corpus arabe *LABR_ar*. Ce corpus arabe sans bruit est noté *LABR_ar_sans-bruit*. Le classifieur RL est à nouveau entraîné sur *LABR_ar_sans-bruit*. Nous espérons que ce traitement aurait un impact positif sur les performances. Malheureusement, nous avons obtenu une exactitude de 73.1% sur *LABR_ar_sans-bruit* qui est inférieure à celle (74.6%) obtenue sur *LABR_ar*. De cette façon, les mots bruit semblent être informatifs dans le corpus en arabe, même s'ils ne le sont pas pour le corpus en anglais.

Nous avons trouvé, dans la section B.4.1, que l'application du prétraitement du corpus en *LABR_ar* donne une performance égale à 76.7%. Cette dernière est supérieure à 74.6% obtenu par le système *AOA-baseline* sur le corpus *LABR_ar*. Cela signifie que l'étape de prétraitement de type light stemming est importante pour l'analyse d'opinions en arabe. De plus, cette performance (76.7%) est légérement supérieure à 76.3% obtenue sur le corpus *LABR_en*. Nous en déduisons que la traduction automatique, en tant qu'outil statistique, et le light stemming, en tant qu'outil linguistique, permettent d'opbtenir des résultats proches pour la tâche d'analyse d'opinions en arabe. Donc, pour les langues sans ces outils linguistiques, nous pourrions appliquer la traduction automatique pour avoir un bon

	Ensembles	Exactitude
Changement du domaine	Train = IMDB Dev = LABR-dev Test = LABR-test	70.6%
Multi-Domaines	Train = IMDB + LABR-train Dev = LABR-dev Test = LABR-test	73.1%
	Train = LABR-train Dev = LABR-dev Test = LABR-test	**75.1%**

TABLE B.6 – Configurations et résultats des expériences réalisées pour élargir le corpus d'apprentissage

système d'analyse d'opinions, en particulier avec les progrès du domaine de la traduction automatique. En d'autres termes, nous pourrions utiliser la traduction automatique si l'outil linguistique spécifique n'est pas encore développé ou n'est pas performant.

Pour aller plus loin dans l'exploration de l'impact des systèmes de traduction automatique dans l'AO, nous pensons que les performances obtenues avec le système *AOA-MT* peuvent être améliorées en utilisant une plus grande quantité de données d'apprentissage. De plus, nous pensons qu'un plus grand corpus, quel que soit le domaine, peut améliorer les performances. Dans cette perspective, nous avons fait quelques expériences en faisant varier le domaine (livres, films) des critiques. Les configurations que nous avons testées sont décrites ci-dessous :

— Le corpus anglais IMDB Maas *et al.* [2011] utilisé comme ensemble d'apprentissage *Train*.
— Ensemble d'apprentissage *train* composé de la fusion du corpus IMDB et de l'ensemble d'apprentissage du LABR.
— Ensemble d'apprentissage composé de ensemble d'apprentissage LABR avec utilisation du modèle du classification qui est initialisé avec les paramètres obtenus après l'apprentissage du classifieur sur le corpus IMDB.

Les résultats sont rapportés dans le tableau B.6. Nous observons qu'en changeant le domaine de l'ensemble d'apprentissage, l'exactitude passe de 76.3% à 70.6% : une baisse de 5 points environ. Cependant, lors de l'ajout d'un ensemble d'exemples appartenant à un autre domaine à l'ensemble d'apprentissage, nous obtenons 73.1% comme exactitude. Ainsi, fusionner les domaines dans le corpus d'apprentissage fonctionne mieux que changer complètement le domaine d'apprentissage (73.1% vs. 70.6%). En plus de la fusion de différents domaines, une expérience multi-domaines peut également être établie en initialisant les paramètres du classifieur avec ceux obtenus lors de l'apprentissage sur l'ensemble d'exemples dans un autre domaine. L'exactitude atteint 75.1% avec cette technique. C'est mieux que de s'entraîner sur des données mixtes appartenant à des domaines différents (75.1% vs 73.1%).

B.5 Conclusion

Dans ce travail, nous avons présenté un ensemble d'expériences pour étudier l'impact de la traduction automatique en anglais sur l'analyse d'opinions en arabe. Nos expériences montrent que l'AO sur le corpus traduit en anglais est meilleure que celle sur le corpus brut en arabe. Nous avons observé que la traduction automatique ne modifie pas la détection de la polarité. De plus, nous avons constaté que l'AO sur le corpus traduit en anglais atteint des résultats compétitifs par rapport à l'AO des textes en arabes prétraités avec un light stemming. Ainsi, nous pourrions généraliser que, quelle que soit la langue, la traduction automatique puisse être utilisée si de tels outils linguistiques (light stemming) n'existent pas ou ne sont pas performants. Nous avons également exploré la voie de l'extension du corpus d'apprentissage et nous avons prouvé l'intérêt de conserver, dans le corpus d'apprentissage, des données dont le domaine est celui de l'ensemble de test. Nous avons également montré que le mélange de données d'apprentissage appartenant à plusieurs domaines est plus performant qu'un simple changement du domaine des données d'apprentissage.

Ce travail a été réalisé au début de la thèse. Les systèmes utilisés sont simples et classiques. Il est préferable d'approffondir ce travail préliminaire. Plusieurs pistes sont envisageables :

— tester le réseau convolutif CNN sur le corpus traduit en anglais.

— utiliser les ressources disponibles en analyse d'opinions en anglais

Une autre perspective consiste à combiner le systéme basé sur les embeddings spécifiques à l'arabe (dans le chapitre 5) et le système basé sur la traduction. Le protocole de combinaison peut être de façon séquentielle ou paralléle.

Bibliographie personnelle

Amira Barhoumi, Vincent Levorato, Nicolas Dugué, Nathalie Camelin. « "L'important c'est de participer" : positive ironie. Analyse de sentiments et détection de l'ironie Les systèmes du LIUM et d'OCTO ». (DEFT 2017), Jun 2017, Orléans, France.

Amira Barhoumi, Yannick Estève, Chafik Aloulou, Lamia Belguith. «Document embeddings for Arabic Sentiment Analysis». International conference on Language Processing and Knowledge Management, (LPKM 2017), Sep 2017, Sfax, Tunisia.

Amira Barhoumi, Nathalie Camelin, Yannick Estève. « Des représentations continues de mots pour l'analyse d'opinions en arabe : une étude qualitative ». 25e conférence sur le Traitement Automatique des Langues Naturelles (TALN 2018), May 2018, Rennes, France. Actes de la conférence TALN 2018.

Amira Barhoumi, Chafik Aloulou, Nathalie Camelin, Yannick Estève, Lamia Belguith. «Arabic Sentiment analysis : an empirical study of machine translation's impact». International conference on Language Processing and Knowledge Management, (LPKM2018), Oct 2018, Sfax, Tunisia.

Amira Barhoumi, Nathalie Camelin, Chafik Aloulou, Yannick Estève, Lamia Belguith. «embeddings spécifiques à la langue arabe : application à l'analyse d'opinions Traitement Automatique des Langues Naturelles (TALN 2019), Toulouse, France.

Amira Barhoumi, Nathalie Camelin, Chafik Aloulou, Yannick Estève, Lamia Hadrich Belguith. «An Empirical Evaluation of Arabic-Specific Embeddings for Sentiment Analysis ». International Conference on Arabic Language Processing, Oct 2019, Nancy, France. pp.34-48.

Amira Barhoumi, Nathalie Camelin, Chafik Aloulou, Yannick Estève, Lamia Hadrich Belguith. « Toward Qualitative Evaluation of Embeddings for Arabic Sentiment Analysis ». International Conference on Language Resources and Evaluation (LREC2020), May 2020, Marseille, France.

Bibliographie

AABED M. A., AWAIDEH S. M., ELSHAFEI A.-R. M. & GUTUB A. A. (2007). Arabic diacritics based steganography. In *2007 IEEE International Conference on Signal Processing and Communications*, p. 756–759 : IEEE.

ABBASI A., CHEN H. & SALEM A. (2008). Sentiment analysis in multiple languages : Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)*, **26**(3), 12.

ABDAOUI A., NZALI M. D. T., AZÉ J., BRINGAY S., LAVERGNE C., MOLLEVI C. & PONCELET P. (2015). Advanse : Sentiment, opinion and emotion analysis in french tweets.

ABDELALI A., DARWISH K., DURRANI N. & MUBARAK H. (2016). Farasa : A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics : Demonstrations*, p. 11–16.

ABDELHADE N., SOLIMAN T. H. A. & IBRAHIM H. M. (2017). Detecting twitter users' opinions of arabic comments during various time episodes via deep neural network. In *International Conference on Advanced Intelligent Systems and Informatics*, p. 232–246 : Springer.

ABDUL-MAGEED M. & DIAB M. (2012a). Toward building a large-scale arabic sentiment lexicon. In *Proceedings of the 6th international global WordNet conference*, p. 18–22.

ABDUL-MAGEED M., DIAB M. & KÜBLER S. (2014). Samar : Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, **28**(1), 20–37.

ABDUL-MAGEED M. & DIAB M. T. (2012b). Awatif : A multi-genre corpus for modern standard arabic subjectivity and sentiment analysis. In *LREC*, volume 515, p. 3907–3914 : Citeseer.

ABDUL-MAGEED M. & DIAB M. T. (2014). Sana : A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis. In *LREC*, p. 1162–1169.

ABDUL-MAGEED M., DIAB M. T. & KORAYEM M. (2011). Subjectivity and sentiment analysis of modern standard arabic. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : Short Papers - Volume 2*, HLT '11, p. 587–591, Stroudsburg, PA, USA : Association for Computational Linguistics.

ABDULLA N., MOHAMMED S., AL-AYYOUB M., AL-KABI M. *et al.* (2014a). Automatic lexicon construction for arabic sentiment analysis. In *2014 International Conference on Future Internet of Things and Cloud*, p. 547–552 : IEEE.

ABDULLA N. A., AHMED N. A., SHEHAB M. A. & AL-AYYOUB M. (2013). Arabic sentiment analysis : Lexicon-based and corpus-based. In *2013 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT)*, p. 1–6 : IEEE.

ABDULLA N. A., AHMED N. A., SHEHAB M. A., AL-AYYOUB M., AL-KABI M. N. & AL-RIFAI S. (2014b). Towards improving the lexicon-based approach for arabic sentiment analysis. *International Journal of Information Technology and Web Engineering (IJITWE)*, **9**(3), 55–71.

ABUAIADH D. (2013). Dataset for arabic document classification. 2013.

AL-AYYOUB M., KHAMAISEH A. A., JARARWEH Y. & AL-KABI M. N. (2019). A comprehensive survey of arabic sentiment analysis. *Information Processing & Management*, **56**(2), 320–342.

AL-AYYOUB M., NUSEIR A., KANAAN G. & AL-SHALABI R. (2016). Hierarchical classifiers for multi-way sentiment analysis of arabic reviews. *International Journal of Advanced Computer Science and Application*, **7**(2).

AL-AZANI S. & EL-ALFY E.-S. M. (2017). Hybrid deep learning for sentiment polarity determination of arabic microblogs. In *International Conference on Neural Information Processing*, p. 491–500 : Springer.

AL-HORAIBI L. & KHAN M. B. (2016). Sentiment analysis of arabic tweets using semantic resources. *International Journal of Computing & Information Sciences*, **12**(2), 149.

AL-KABI M., AL-AYYOUB M., ALSMADI I. & WAHSHEH H. (2016). A prototype for a standard arabic sentiment analysis corpus. *Int. Arab J. Inf. Technol.*, **13**(1A), 163–170.

AL-KABI M. N., GIGIEH A. H., ALSMADI I. M., WAHSHEH H. A. & HAIDAR M. M. (2014). Opinion mining and analysis for arabic language. *International Journal of Advanced Computer Science and Applications (IJACSA), SAI Publisher*, **5**(5), 181–195.

AL-MOSLMI T., ALBARED M., AL-SHABI A., OMAR N. & ABDULLAH S. (2018). Arabic senti-lexicon : Constructing publicly available language resources for arabic sentiment analysis. *Journal of Information Science*, **44**(3), 345–362.

AL-OSAIMI S. & BADRUDDIN K. M. (2014). Role of emotion icons in sentiment classification of arabic tweets. In *Proceedings of the 6th international conference on management of emergent digital ecosystems*, p. 167–171 : ACM.

AL-ROWAILY K., ABULAISH M., HALDAR N. A.-H. & AL-RUBAIAN M. (2015). Bisal–a bilingual sentiment analysis lexicon to analyze dark web forums for cyber security. *Digital Investigation*, **14**, 53–62.

AL-SALLAB A., BALY R., HAJJ H., SHABAN K. B., EL-HAJJ W. & BADARO G. (2017). Aroma : A recursive deep learning model for opinion mining in arabic as a low resource language. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, **16**(4), 25.

AL SALLAB A., HAJJ H., BADARO G., BALY R., EL HAJJ W. & SHABAN K. B. (2015). Deep learning models for sentiment analysis in arabic. In *Proceedings of the second workshop on Arabic natural language processing*, p. 9–17.

AL-SHABI A., ADEL A., OMAR N. & AL-MOSLMI T. (2017). Cross-lingual sentiment classification from english to arabic using machine translation. *International journal of advanced computer science and applications*, **8**(12), 434–440.

AL-THUBAITY A., ALHARBI M., ALQAHTANI S. & ALJANDAL A. (2018). A saudi dialect twitter corpus for sentiment and emotion analysis. In *2018 21st Saudi Computer Society National Computer Conference (NCC)*, p. 1–6 : IEEE.

AL-TWAIRESH N., AL-KHALIFA H., AL-SALMAN A. & AL-OHALI Y. (2017). Arasenti-tweet : A corpus for arabic sentiment analysis of saudi tweets. *Procedia Computer Science*, **117**, 63–72.

AL-TWAIRESH N., AL-KHALIFA H. & ALSALMAN A. (2016). Arasenti : large-scale twitter-specific arabic sentiment lexicons. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, volume 1, p. 697–705.

ALAYBA A. M., PALADE V., ENGLAND M. & IQBAL R. (2017). Arabic language sentiment analysis on health services. In *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, p. 114–118 : IEEE.

ALAYBA A. M., PALADE V., ENGLAND M. & IQBAL R. (2018a). A combined cnn and lstm model for arabic sentiment analysis. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, p. 179–191 : Springer.

ALAYBA A. M., PALADE V., ENGLAND M. & IQBAL R. (2018b). Improving sentiment analysis in arabic using word representation. In *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*, p. 13–18 : IEEE.

ALDAYEL H. K. & AZMI A. M. (2016). Arabic tweets sentiment analysis–a hybrid scheme. *Journal of Information Science*, **42**(6), 782–797.

ALHUMOUD S., ALBUHAIRI T. & ALOHAIDEB W. (2015a). Hybrid sentiment analyser for arabic tweets using r. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, volume 1, p. 417–424 : IEEE.

ALHUMOUD S., ALBUHAIRI T. & ALTUWAIJRI M. (2015b). Arabic sentiment analysis using weka a hybrid learning approach. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, volume 1, p. 402–408 : IEEE.

ALMAS Y. & AHMAD K. (2007). A note on extracting 'sentiments' in financial news in english, arabic & urdu. In *The Second Workshop on Computational Approaches to Arabic Script-based Languages*, p. 1–12.

ALNAWAS A. & ARICI N. (2018). The corpus based approach to sentiment analysis in modern standard arabic and arabic dialects : A literature review. *Politeknik Dergisi*, **21**(2), 461–470.

ALOTAIBI S. S. & ANDERSON C. W. (2016). Extending the knowledge of the arabic sentiment classification using a foreign external lexical source. *Int. J. Nat. Lang. Comput*, **5**(3), 1–11.

ALOTAIBY F., FODA S. & ALKHARASHI I. (2010). Clitics in arabic language : a statistical study. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, p. 595–601.

ALSHARGI F., DIBAS S., ALKHEREYF S., FARAJ R., ABDULKAREEM B., YAGI S., KACHA O., HABASH N. & RAMBOW O. (2019). Morphologically annotated corpora for seven arabic dialects : Taizi, sanaani, najdi, jordanian, syrian, iraqi and moroccan. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, p. 137–147.

ALTRABSHEH N., GABER M. M. & COCEA M. (2013). Sa-e : sentiment analysis for education. In *International Conference on Intelligent Decision Technologies*, volume 255, p. 353–362.

ALY M. & ATIYA A. (2013). LABR : A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, p. 494–498, Sofia, Bulgaria : Association for Computational Linguistics.

ANDO S. & HUANG C. Y. (2017). Deep over-sampling framework for classifying imbalanced data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, p. 770–785 : Springer.

ANDRASON A. (2016). Left dislocation in arabic : The complexity of form and meaning. *Stellenbosch Papers in Linguistics Plus*, **50**, 111–138.

ANTONIAK M. & MIMNO D. (2018). Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics*, **6**, 107–119.

M. APIDIANAKI, S. M. MOHAMMAD, J. MAY, E. SHUTOVA, S. BETHARD & M. CARPUAT, Eds. (2018). *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana. Association for Computational Linguistics.

ARTSTEIN R. & POESIO M. (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics*, **34**(4), 555–596.

ASSIRI A., EMAM A. & AL-DOSSARI H. (2016). Saudi twitter corpus for sentiment analysis. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, **10**(2), 272–275.

ASSIRI A., EMAM A. & AL-DOSSARI H. (2018). Towards enhancement of a lexicon-based approach for saudi dialect sentiment analysis. *Journal of Information Science*, **44**(2), 184–202.

ATTIA M. & SOMERS H. (2008). *Handling Arabic morphological and syntactic ambiguity within the LFG framework with a view to machine translation*, volume 279. University of Manchester Manchester.

BACCOUCHE T. (1974). Esquisse d'une étude comparative des schémas des verbes en arabe classique et en arabe tunisien. *Les cahiers de Tunisie*, **22**, 87–88.

BACCOUCHE T. (1994). *L'emprunt en arabe moderne*. Académie tunisienne des sciences, des lettres, et des arts, Beït al-Hikma.

BADACHE I. (2019). Users' traces for enhancing arabic facebook search. In *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, p. 241–245 : ACM.

BADACHE I., ABU-THAHER A., HAMDAN M. & ABU-JAISH L. (2019). Social Information Retrieval in Arabic Language : Case of Facebook. In *COnférence en Recherche d'Information et Applications*, Lyon, France.

BADARO G., BALY R., HAJJ H., HABASH N. & EL-HAJJ W. (2014a). A large scale Arabic sentiment lexicon for Arabic opinion mining. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, p. 165–173, Doha, Qatar : Association for Computational Linguistics.

BADARO G., BALY R., HAJJ H., HABASH N. & EL-HAJJ W. (2014b). A large scale arabic sentiment lexicon for arabic opinion mining. In *Proceedings of the EMNLP 2014 workshop on arabic natural language processing (ANLP)*, p. 165–173.

BADARO G., JUNDI H., HAJJ H., EL-HAJJ W. & HABASH N. (2018). Arsel : A large scale arabic sentiment and emotion lexicon. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Paris, France. European Language Resources Association (ELRA)*.

BALY R., BADARO G., EL-KHOURY G., MOUKALLED R., AOUN R., HAJJ H., EL-HAJJ W., HABASH N. & SHABAN K. (2017a). A characterization study of arabic twitter data with a benchmarking for state-of-the-art opinion mining models. In *Proceedings of the third Arabic natural language processing workshop*, p. 110–118.

BALY R., HAJJ H., HABASH N., SHABAN K. B. & EL-HAJJ W. (2017b). A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in arabic. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, **16**(4), 23.

BALY R., KHADDAJ A., HAJJ H., EL-HAJJ W. & SHABAN K. B. (2019). Arsentd-lev : A multi-topic corpus for target-based sentiment analysis in arabic levantine tweets. *arXiv preprint arXiv :1906.01830*.

BARONI M., DINU G. & KRUSZEWSKI G. (2014). Don't count, predict ! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 238–247.

BAYOUDHI A., GHORBEL H. & BELGUITH L. H. (2015). Sentiment classification of arabic documents : Experiments with multi-type features and ensemble algorithms. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, p. 196–205.

BENAMARA F., CESARANO C., PICARIELLO A., RECUPERO D. R. & SUBRAHMANIAN V. S. (2007). Sentiment analysis : Adjectives and adverbs are better than adjectives alone. In *ICWSM*, p. 1–7 : Citeseer.

BENAMARA F., GROUIN C., KAROUI J., MORICEAU V. & ROBBA I. (2017). Analyse d'opinion et langage figuratif dans des tweets : présentation et résultats du défi fouille de textes deft2017.

BENGIO Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks : Tricks of the trade*, p. 437–478. Springer.

BENGIO Y., ROUX N. L., VINCENT P., DELALLEAU O. & MARCOTTE P. (2006). Convex neural networks. In *Advances in neural information processing systems*, p. 123–130.

BENGIO Y., SIMARD P., FRASCONI P. *et al.* (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, **5**(2), 157–166.

S. BETHARD, M. CARPUAT, M. APIDIANAKI, S. M. MOHAMMAD, D. CER & D. JURGENS, Eds. (2017). *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada. Association for Computational Linguistics.

S. BETHARD, M. CARPUAT, D. CER, D. JURGENS, P. NAKOV & T. ZESCH, Eds. (2016). *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.

BOJANOWSKI P., GRAVE E., JOULIN A. & MIKOLOV T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*.

BOJANOWSKI P., GRAVE E., JOULIN A. & MIKOLOV T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, **5**, 135–146.

BOLOGNA G. & HAYASHI Y. (2018). A rule extraction study from svm on sentiment analysis. *Big Data and Cognitive Computing*, **2**(1), 6.

BOSCO C., PATTI V. & BOLIOLI A. (2013). Developing corpora for sentiment analysis : The case of irony and senti-tut. *IEEE intelligent systems*, **28**(2), 55–63.

BOTTOU L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, p. 177–186. Springer.

BOUDAD N., FAIZI R., OULAD HAJ THAMI R. & CHIHEB R. (2017). Sentiment analysis in arabic : A review of the literature. *Ain Shams Eng J (2017), http ://dx.doi.org/10.1016/j.asej.2017.04.007*.

BOUJELBANE R., ELLOUZE M., BÉCHET F. & BELGUITH L. (2015). De l'arabe standard vers l'arabe dialectal : projection de corpus et ressources linguistiques en vue du traitement automatique de l'oral dans les médias tunisiens.

BOUKADIDA N. (2008). *Connaissances phonologiques et morphologiques dérivationnelles et apprentissage de la lecture en arabe (Etude longitudinale)*. PhD thesis.

BRANCO P., TORGO L. & RIBEIRO R. (2015). A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv :1505.01658*.

CAN E. F., EZEN-CAN A. & CAN F. (2018). Multilingual sentiment analysis : An rnn-based framework for limited data. *arXiv preprint arXiv :1806.04511*.

CHAWLA N. V., BOWYER K. W., HALL L. O. & KEGELMEYER W. P. (2002). Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, **16**, 321–357.

CHIU J. P. & NICHOLS E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, **4**, 357–370.

CHO K., VAN MERRIËNBOER B., BAHDANAU D. & BENGIO Y. (2014). On the properties of neural machine translation : Encoder-decoder approaches. *arXiv preprint arXiv :1409.1259*.

CLARK E. M. (2019). Applications in sentiment analysis and machine learning for identifying public health variables across social media.

CLICHE M. (2017). Bb_twtr at semeval-2017 task 4 : twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv :1704.06125*.

COHEN D. (1970). Essai d'une analyse automatique de l'arabe. *Etudes de linguistique sémitique et arabe*, p. 49–78.

COLLOBERT R. & BENGIO S. (2004). Links between perceptrons, mlps and svms. In *Proceedings of the twenty-first international conference on Machine learning*, p.23 : ACM.

COLLOBERT R. & WESTON J. (2008). A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, p. 160–167 : ACM.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011a). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, **12**(Aug), 2493–2537.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011b). Natural language processing (almost) from scratch. *Journal of machine learning research*, **12**(Aug), 2493–2537.

CROCE D., CASTELLUCCI G. & BASILI R. (2016). Injecting sentiment information in context-aware convolutional neural networks. In *IIR*.

DAHOU A., ELAZIZ M. A., ZHOU J. & XIONG S. (2019). Arabic sentiment classification using convolutional neural network and differential evolution algorithm. *Computational intelligence and neuroscience*, **2019**.

DAHOU A., XIONG S., ZHOU J., HADDOUD M. H. & DUAN P. (2016). Word embeddings and convolutional neural network for arabic sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, p. 2418–2427.

DAI A. M. & LE Q. V. (2015). Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, p. 3079–3087.

DAI W., DAI C., QU S., LI J. & DAS S. (2017). Very deep convolutional neural networks for raw waveforms. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 421–425 : IEEE.

DAS S. & CHEN M. (2001). Yahoo ! for amazon : Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific finance association annual conference (APFA)*, volume 35, p.43 : Bangkok, Thailand.

DEAN J., CORRADO G., MONGA R., CHEN K., DEVIN M., MAO M., SENIOR A., TUCKER P., YANG K., LE Q. V. *et al.* (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, p. 1223–1231.

DEBILI F. & ACHOUR H. (1998). Voyellation automatique de l'arabe. In *Computational Approaches to Semitic Languages*.

DEBILI F., ACHOUR H. & SOUISSI E. (2002). La langue arabe et l'ordinateur : de l'étiquetage grammatical à la voyellation automatique. *Correspondances*, **71**, 10–28.

DESHPANDE A. (2016). A beginner's guide to understanding convolutional neural networks. *Retrieved March*, **31**(2017).

DESJARDINS J. (2005). L'analyse de régression logistique. *Tutorial in quantitative methods for psychology*, **1**(1), 35–41.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

DHAOUI C., WEBSTER C. M. & TAN L. P. (2017). Social media sentiment analysis : lexicon versus machine learning. *Journal of Consumer Marketing*, **34**(6), 480–488.

DICHY J. (1990). *L'écriture dans la représentation de la langue : la lettre et le mot en arabe*. PhD thesis, Lyon 2.

DICHY J. (1997). Pour une lexicomatique de l'arabe : l'unité lexicale simple et l'inventaire fini des spécificateurs du domaine du mot. *Meta : journal des traducteurs/Meta : Translators' Journal*, **42**(2), 291–306.

DOZAT T. (2016). Incorporating nesterov momentum into adam.

DUCHI J., HAZAN E. & SINGER Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**(Jul), 2121–2159.

DUWAIRI R., AHMED N. A. & AL-RIFAI S. Y. (2015). Detecting sentiment embedded in arabic social media–a lexicon-based approach. *Journal of Intelligent & Fuzzy Systems*, **29**(1), 107–117.

DUWAIRI R. M. (2015). Sentiment analysis for dialectical arabic. In *2015 6th International Conference on Information and Communication Systems (ICICS)*, p. 166–170 : IEEE.

EL-BELTAGY S. R. (2017). Weightednileulex : A scored arabic sentiment lexicon for improved sentiment analysis. *Language Processing, Pattern Recognition and Intelligent Systems. Special Issue on Computational Linguistics, Speech& Image Processing for Arabic Language. World Scientific Publishing Co*.

EL-BELTAGY S. R. & ALI A. (2013). Open issues in the sentiment analysis of arabic social media : A case study. In *2013 9th International Conference on Innovations in Information Technology (IIT)*, p. 215–220 : IEEE.

EL-HALEES A. (2011). Arabic opinion mining using combined. *Proceeding the International Arab Conference On Information Technology*.

EL-KHAIR I. A. (2016). 1.5 billion words arabic corpus. *arXiv preprint arXiv :1611.04033*.

EL MAHDAOUY A., EL ALAOUI S. O. & GAUSSIER E. (2018). Improving arabic information retrieval using word embedding similarities. *Int. J. Speech Technol.*, **21**(1), 121–136.

ELKAN C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, p. 973–978 : Lawrence Erlbaum Associates Ltd.

ELMAHDY M., GRUHN R. & MINKER W. (2012). *Novel techniques for dialectal arabic speech recognition*. Springer Science & Business Media.

ELMAN J. L. (1990). Finding structure in time. *Cognitive science*, **14**(2), 179–211.

ELNAGAR A. & EINEA O. (2018). Brad 2.0 : Book reviews in arabic dataset. *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, p. 1–8.

ELNAGAR A., EINEA O. & LULU L. (2017). Comparative study of sentiment classification for automated translated latin reviews into arabic. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, p. 443–448.

ELNAGAR A., KHALIFA Y. S. & EINEA A. (2018a). Hotel arabic-reviews dataset construction for sentiment analysis applications. In *Intelligent Natural Language Processing : Trends and Applications*, p. 35–52. Springer.

ELNAGAR A., LULU L. & EINEA O. (2018b). An annotated huge dataset for standard and colloquial arabic reviews for subjective sentiment analysis. *Procedia computer science*, **142**, 182–189.

ELRAZZAZ M., ELBASSUONI S., SHABAN K. & HELWE C. (2017). Methodical evaluation of Arabic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, p. 454–458, Vancouver, Canada : Association for Computational Linguistics.

ELSAHAR H. & EL-BELTAGY S. R. (2014). A fully automated approach for arabic slang lexicon extraction from microblogs. In A. GELBUKH, Ed., *Computational Linguistics and Intelligent Text Processing*, p. 79–91, Berlin, Heidelberg : Springer Berlin Heidelberg.

ELSAHAR H. & EL-BELTAGY S. R. (2015). Building large arabic multi-domain resources for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, p. 23–34 : Springer.

ELSHAKANKERY K. & AHMED M. F. (2019). Hilatsa : A hybrid incremental learning approach for arabic tweets sentiment analysis. *Egyptian Informatics Journal*.

ERWIN W. M. (1973). The problem of diglossia in arabic : A comparative study of classical and iraqi arabic.

ESKANDER R. & RAMBOW O. (2015). Slsa : A sentiment lexicon for standard arabic. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, p. 2545–2550.

ESULI A. & SEBASTIANI F. (2006). Sentiwordnet : A publicly available lexical resource for opinion mining. In *LREC*, volume 6, p. 417–422 : Citeseer.

FARGHALY A. (2010). The arabic language, arabic linguistics and arabic computational linguistics. *Arabic Computational Linguistics (edt. Ali Farghaly), San Francisco : Center for the Study of Language and Inf*, p. 43–81.

FARRA N., CHALLITA E., ASSI R. A. & HAJJ H. (2010). Sentence-level and document-level sentiment mining for arabic texts. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, p. 1114–1119 : IEEE.

FOUAD M. M., MAHANY A., ALJOHANI N., ABBASI R. A. & HASSAN S.-U. (2019). Arwordvec : efficient word embedding models for arabic tweets. *Soft Computing*.

FUKUSHIMA K. (1980). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, **36**(4), 193–202.

GABARRON E., DORRONZORO E., RIVERA-ROMERO O. & WYNN R. (2019). Diabetes on twitter : a sentiment analysis. *Journal of diabetes science and technology*, **13**(3), 439–444.

GAMAL D., ALFONSE M., EL-HORBATY E.-S. M. & SALEM A.-B. M. (2019). Twitter benchmark dataset for arabic sentiment analysis. *International Journal of Modern Education and Computer Science*, **11**(1), 33.

GERS F. A., SCHRAUDOLPH N. N. & SCHMIDHUBER J. (2002). Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, **3**(Aug), 115–143.

GHANNAY S. (2017). *Étude sur les représentations continues de mots appliquées à la détection automatique des erreurs de reconnaissance de la parole*. PhD thesis, Le Mans.

GHANNAY S., FAVRE B., ESTÈVE Y. & CAMELIN N. (2016). Word embedding evaluation and combination. In *10th edition of the Language Resources and Evaluation Conference (LREC 2016)*, Portorož, Slovenia.

GRAFAREND E. W. (2006). *Linear and nonlinear models : fixed effects, random effects, and mixed models*. de Gruyter.

GRAVE E., BOJANOWSKI P., GUPTA P., JOULIN A. & MIKOLOV T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

GRAVES A. & SCHMIDHUBER J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, **18**(5), 602 – 610. IJCNN 2005.

GUELLIL I., ADEEL A., AZOUAOU F., BENALI F., HACHANI A.-E. & HUSSAIN A. (2018a). Arabizi sentiment analysis based on transliteration and automatic corpus annotation. In *Proceedings of the 9th workshop on computational approaches to subjectivity, sentiment and social media Analysis*, p. 335–341.

GUELLIL I., ADEEL A., AZOUAOU F. & HUSSAIN A. (2018b). Sentialg : Automated corpus annotation for algerian sentiment analysis. In *International Conference on Brain Inspired Cognitive Systems*, p. 557–567 : Springer.

GUTMANN M. U. & HYVÄRINEN A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, **13**(Feb), 307–361.

HABASH N., DIAB M. T. & RAMBOW O. (2012). Conventional orthography for dialectal arabic. In *LREC*, p. 711–718.

HABASH N. & RAMBOW O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, p. 573–580.

HABASH N. Y. (2010). Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, **3**(1), 1–187.

HADI W. (2015). Classification of arabic social media data. *Advances in Computational Sciences and Technology*, **8**(1), 29–34.

HAMDI A. (2015). *Traitement automatique du dialecte tunisien à l'aide d'outils et de ressources de l'arabe standard : application à l'étiquetage morphosyntaxique*. PhD thesis, Aix-Marseille.

HAMDI A., BOUJELBANE R., HABASH N. & NASR A. (2013). The effects of factorizing root and pattern mapping in bidirectional tunisian-standard arabic machine translation.

HAMOUDA A. E.-D. A. & EL-TAHER F. E.-Z. (2013). Sentiment analyzer for arabic comments system. *Int. J. Adv. Comput. Sci. Appl*, **4**(3).

HARRIS Z. S. (1954). Distributional structure. *Word*, **10**(2-3), 146–162.

HE H. & MA Y. (2013). *Imbalanced learning : foundations, algorithms, and applications*. John Wiley & Sons.

HEDAR A. R. & DOSS M. (2013). Mining social networks arabic slang comments. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*.

HEIKAL M., TORKI M. & EL-MAKKY N. (2018). Sentiment analysis of arabic tweets using deep learning. *Procedia Computer Science*, **142**, 114–122.

HOCHREITER S., BENGIO Y., FRASCONI P., SCHMIDHUBER J. *et al.* (2001). Gradient flow in recurrent nets : the difficulty of learning long-term dependencies.

HOCHREITER S. & SCHMIDHUBER J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

HOPFIELD J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, **79**(8), 2554–2558.

HU M. & LIU B. (2004a). Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, p. 168–177, New York, NY, USA : ACM.

HU M. & LIU B. (2004b). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 168–177 : ACM.

HUBEL D. H. & WIESEL T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, **160**(1), 106–154.

HUSSEIN D. M. E.-D. M. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, **30**(4), 330–338.

IBRAHIM H. S., ABDOU S. M. & GHEITH M. (2015a). Automatic expandable large-scale sentiment lexicon of modern standard arabic and colloquial. In *Arabic Computational Linguistics (ACLing), 2015 First International Conference on*, p. 94–99 : IEEE.

IBRAHIM H. S., ABDOU S. M. & GHEITH M. (2015b). Idioms-proverbs lexicon for modern standard arabic and colloquial sentiment analysis. *arXiv preprint arXiv :1506.01906*.

IBRAHIM H. S., ABDOU S. M. & GHEITH M. (2015c). Sentiment analysis for modern standard arabic and colloquial. *arXiv preprint arXiv :1505.03105*.

IBRAHIM H. S., ABDOU S. M. & GHEITH M. (2015d). Sentiment analysis for modern standard arabic and colloquial. *International Journal on Natural Language Computing (IJNLC)*, **4**(2).

IBRAHIM M. A. & SALIM N. (2013). Opinion analysis for twitter and arabic tweets : A systematic literature review. *Journal of Theoretical & Applied Information Technology*, **56**(3).

INÈS S. (2005). La morphologie des noms de professions : incorporation et paraphrase. *BOUHLEL Ezzedine :" La fonction sujet et les paradigmes afférents en français*, p. 156.

IRSOY O. & CARDIE C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, p. 720–728.

ITANI M., ROAST C. & AL-KHAYATT S. (2017). Corpora for sentiment analysis of arabic text in social media. In *2017 8th international conference on information and communication systems (ICICS)*, p. 64–69 : IEEE.

ITANI M. M., ZANTOUT R. N., HAMANDI L. & ELKABANI I. (2012). Classifying sentiment in arabic social networks : Naive search versus naive bayes. In *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, p. 192–197 : IEEE.

JAAFAR Y., NAMLY D., BOUZOUBAA K. & YOUSFI A. (2017). Enhancing arabic stemming process using resources and benchmarking tools. *Journal of King Saud University - Computer and Information Sciences*, **29**(2), 164 – 170. Arabic Natural Language Processing : Models, Systems and Applications.

JAMAL M., BENATIA E., ELYAAKOUBI M. & LAZREK A. (2006). Arabic text justification.

JAMES W. (1884). What is an emotion ? *Mind*, **9**(34), 188–205.

JARRAR M., HABASH N., AKRA D. F. & ZALMOUT N. (2014). Building a corpus for palestinian arabic : a preliminary study.

JORDAN M. I. (1997). Serial order : A parallel distributed processing approach. In *Advances in psychology*, volume 121, p. 471–495. Elsevier.

JOULIN A., GRAVE E. & MIKOLOV P. B. T. (2017). Bag of tricks for efficient text classification. *EACL 2017*, p. 427.

KAMIR D., SOREQ N. & NEEMAN Y. (2002). A comprehensive nlp system for modern standard arabic and modern hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, p. 1–9 : Association for Computational Linguistics.

KAUR A. & GUPTA V. (2013). A survey on sentiment analysis and opinion mining techniques. *Journal of Emerging Technologies in Web Intelligence*, **5**(4), 367–371.

KEES V. (1997). The arabic language.

KHALIFA K. & OMAR N. (2014). A hybrid method using lexicon-based approach and naive bayes classifier for arabic opinion question answering. *JCS*, **10**(10), 1961–1968.

KIM Y. (2014a). Convolutional neural networks for sentence classification. *arXiv preprint arXiv :1408.5882*.

KIM Y. (2014b). Convolutional neural networks for sentence classification. In *In EMNLP* : Citeseer.

KINGMA D. P. & BA J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

KIRITCHENKO S., ZHU X. & MOHAMMAD S. M. (2014). Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, **50**, 723–762.

KOEHN P., HOANG H., BIRCH A., CALLISON-BURCH C., FEDERICO M., BERTOLDI N., COWAN B., SHEN W., MORAN C., ZENS R. *et al.* (2007). Moses : Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, p. 177–180 : Association for Computational Linguistics.

LACHRAF R., BILLAH NAGOUDI E. M., AYACHI Y., ABDELALI A. & SCHWAB D. (2019). ArbEngVec : Arabic-English Cross-Lingual Word Embedding Model. In *The Fourth Arabic Natural Language Processing Workshop*, Florence, Italy.

LARKEY L. S., BALLESTEROS L. & CONNELL M. E. (2007). Light stemming for arabic information retrieval. In *Arabic computational morphology*, p. 221–243. Springer.

LAROUSSI S. & BOUROUBA Y. (2018). Une approche de vocalisation automatique de texte arabe non vocalisé.

LAWRENCE S., BURNS I., BACK A., TSOI A. C. & GILES C. L. (1998). Neural network classification and prior class probabilities. In *Neural networks : tricks of the trade*, p. 299–313. Springer.

LE Q. & MIKOLOV T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, p. II–1188–II–1196 : JMLR.org.

LECUN Y., BOSER B. E., DENKER J. S., HENDERSON D., HOWARD R. E., HUBBARD W. E. & JACKEL L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, p. 396–404.

LECUN Y., KAVUKCUOGLU K. & FARABET C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, p. 253–256 : IEEE.

LIN Y., SHEN S., LIU Z., LUAN H. & SUN M. (2016). Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 2124–2133.

LIU B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, **5**(1), 1–167.

LIZA F. F. & GRZES M. (2016). An improved crowdsourcing based evaluation technique for word embedding methods. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, p. 55–61.

MA X. & HOVY E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, volume 1, p. 1064–1074.

MAAMOURI M., BIES A., BUCKWALTER T. & MEKKI W. (2004). The penn arabic treebank : Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, p. 466–467 : Cairo.

MAAS A. L., DALY R. E., PHAM P. T., HUANG D., NG A. Y. & POTTS C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies-Volume 1*, p. 142–150 : Association for Computational Linguistics.

MAGHFOUR M. & ELOUARDIGHI A. (2018). Standard and dialectal arabic text classification for sentiment analysis. In *International Conference on Model and Data Engineering*, p. 282–291 : Springer.

MAHYOUB F. H., SIDDIQUI M. A. & DAHAB M. Y. (2014). Building an arabic sentiment lexicon using semi-supervised learning. *Journal of King Saud University-Computer and Information Sciences*, **26**(4), 417–424.

MEDHAFFAR S., BOUGARES F., ESTÈVE Y. & HADRICH-BELGUITH L. (2017). Sentiment analysis of tunisian dialects : Linguistic ressources and experiments. In *Proceedings of the third Arabic natural language processing workshop*, p. 55–61.

MEDSKER L. & JAIN L. C. (1999). *Recurrent neural networks : design and applications*. CRC press.

MEJRI S., SAID M. & SFAR I. (2009). Pluringuisme et diglossie en tunisie. *Synergies Tunisie*, **1**, 53–74.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*.

MIKOLOV T., KOMBRINK S., BURGET L., ČERNOCKỲ J. & KHUDANPUR S. (2011). Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 5528–5531 : IEEE.

MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. S. & DEAN J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, p. 3111–3119.

MINSKY M. (1969). S. papert, perceptrons. *MIT P ress, Cambridge, MA*, **1**(969), 2.

MOBARZ H., RASHOWN M. & FARAG I. (2014). Using automated lexical resources in arabic sentence subjectivity. *International Journal of Artificial Intelligence & Applications*, **5**(6), 1.

MODAVE F., ZHAO Y., KRIEGER J., HE Z., GUO Y., HUO J., PROSPERI M. & BIAN J. (2019). Understanding perceptions and attitudes in breast cancer discussions on twitter.

MOHAMMAD S. & TURNEY P. D. (2013). Crowdsourcing a word-emotion association lexicon. *CoRR*, **abs/1308.6297**.

MOHAMMAD S. M., KIRITCHENKO S. & ZHU X. (2013). Nrc-canada : Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv :1308.6242*.

MOHAMMAD S. M., SALAMEH M. & KIRITCHENKO S. (2016). How translation alters sentiment. *Journal of Artificial Intelligence Research*, **55**, 95–130.

MOHAMMAD S. M. & TURNEY P. D. (2010). Emotions evoked by common words and phrases : Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, p. 26–34 : Association for Computational Linguistics.

MOHAMMAD S. M. & YANG T. W. (2011). Tracking sentiment in mail : How genders differ on emotional axes. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, p. 70–79 : Association for Computational Linguistics.

MOHAMMED A. & KORA R. (2019). Deep learning approaches for arabic sentiment analysis. *Social Network Analysis and Mining*, **9**(1), 52.

MOUNTASSIR A., BENBRAHIM H. & BERRADA I. (2013a). Sentiment classification on arabic corpora. *Document numérique*, **16**(1), 73–96.

MOUNTASSIR A., BENBRAHIM H. & BERRADA I. (2013b). Sentiment classification on arabic corpora : A preliminary cross-study. *Document Numérique*, **16**(1), 73–96.

MOURAD A. & DARWISH K. (2013). Subjectivity and sentiment analysis of modern standard arabic and arabic microblogs. In *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, p. 55–64.

MUNEZERO M., MONTERO C. S., MOZGOVOY M. & SUTINEN E. (2013). Exploiting sentiment analysis to track emotions in students' learning diaries. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, Koli Calling '13, p. 145–152, New York, NY, USA : ACM.

NABIL M., ALY M. & ATIYA A. (2014). Labr : A large scale arabic sentiment analysis benchmark. *arXiv preprint arXiv :1411.6718*.

NABIL M., ALY M. & ATIYA A. (2015). Astd : Arabic sentiment tweets dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 2515–2519.

NAKAGAWA T., INUI K. & KUROHASHI S. (2010). Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, p. 786–794 : Association for Computational Linguistics.

P. NAKOV & T. ZESCH, Eds. (2014). *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland. Association for Computational Linguistics.

P. NAKOV, T. ZESCH, D. CER & D. JURGENS, Eds. (2015). *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado. Association for Computational Linguistics.

NAYAK N., ANGELI G. & MANNING C. D. (2016). Evaluating word embeddings using a representative suite of practical tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, p. 19–23.

OMARA E., MOSA M. & ISMAIL N. (2018). Deep convolutional network for arabic sentiment analysis. In *2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC)*, p. 155–159 : IEEE.

PALAZ D., DOSS M. M. & COLLOBERT R. (2015). Convolutional neural networks-based continuous speech recognition using raw speech signal. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 4295–4299 : IEEE.

PANG B., LEE L. & VAITHYANATHAN S. (2002). Thumbs up ? : sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, p. 79–86 : Association for Computational Linguistics.

PENNINGTON J., SOCHER R. & MANNING C. (2014). Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, p. 1532–1543.

PEREIRA F., GERSHMAN S., RITTER S. & BOTVINICK M. (2016). A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data. *Cognitive neuropsychology*, **33**(3-4), 175–190.

PEREZ-MUNUZURI V., PEREZ-VILLAR V. & CHUA L. O. (1993). Autowaves for image processing on a two-dimensional cnn array of excitable nonlinear circuits : flat and wrinkled labyrinths. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, **40**(3), 174–181.

PETERS M. E., NEUMANN M., IYYER M., GARDNER M., CLARK C., LEE K. & ZETTLEMOYER L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

PIERREJEAN B. & TANGUY L. (2018). Towards qualitative word embeddings evaluation : Measuring neighbors variation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Student Research Workshop*, p. 32–39, New Orleans, Louisiana, USA : Association for Computational Linguistics.

PLAUT D. C. *et al.* (1986). Experiments on learning by back propagation.

PONTIKI M., GALANIS D., PAVLOPOULOS J., PAPAGEORGIOU H., ANDROUTSOPOULOS I. & MANANDHAR S. (2014). SemEval-2014 task 4 : Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, p. 27–35, Dublin, Ireland : Association for Computational Linguistics.

PRABOWO R. & THELWALL M. (2009). Sentiment analysis : A combined approach. *Journal of Informetrics*, **3**(2), 143–157.

PUPIER P. (1998). Une première systématique des évaluatifs en français. *Revue québécoise de linguistique*, **26**(1), 51–78.

PYLIEVA H., CHERNODUB A., GRABAR N. & HAMON T. (2018). Improving Automatic Categorization of Technical vs. Laymen Medical Words using FastText Word Embeddings. In *1st International Workshop on Informatics & Data-Driven Medicine (IDDM 2018)*, Lviv, Ukraine.

QAZANFARI K. & YOUSSEF A. (2019). Word embedding by combining resources and integrating techniques. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, p. 438–443 : IEEE.

RAHAB H., ZITOUNI A. & DJOUDI M. (2019). Sana : sentiment analysis on newspapers comments in algeria. *Journal of King Saud University-Computer and Information Sciences*.

REFAEE E. & RIESER V. (2014a). Can we read emotions from a smiley face ? emoticon-based distant supervision for subjectivity and sentiment analysis of arabic twitter feeds. *EMOTION, SOCIAL SIGNALS, SENTIMENT & LINKED OPEN DATA*, p.5̃1.

REFAEE E. & RIESER V. (2014b). Subjectivity and sentiment analysis of arabic twitter feeds with limited resources. In *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, p.1̃6.

REFAEE E. & RIESER V. (2015). Benchmarking machine translated sentiment analysis for arabic tweets. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Student Research Workshop*, p. 71–78.

REFAEE E. & RIESER V. (2016). ilab-edinburgh at semeval-2016 task 7 : A hybrid approach for determining sentiment intensity of arabic twitter phrases. *Proceedings of SemEval-2016*, p. 474–480.

RILOFF E. & WIEBE J. (2003). Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, p. 105–112.

RISH I. *et al.* (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, p. 41–46.

ROSENBLATT F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

ROSENBLATT F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, **65**(6), 386.

RUMELHART D. E., HINTON G. E., WILLIAMS R. J. *et al.* (1988). Learning representations by back-propagating errors. *Cognitive modeling*, **5**(3), 1.

RUSHDI-SALEH M., MARTÍN-VALDIVIA M. T., UREÑA-LÓPEZ L. A. & PEREA-ORTEGA J. M. (2011a). Bilingual experiments with an arabic-english corpus for opinion mining. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, p. 740–745.

RUSHDI-SALEH M., MARTÍN-VALDIVIA M. T., UREÑA-LÓPEZ L. A. & PEREA-ORTEGA J. M. (2011b). Oca : Opinion corpus for arabic. *Journal of the American Society for Information Science and Technology*, **62**(10), 2045–2054.

SAAD M. K. (2011). Arabic text classification : Text preprocessing, term weighting, and morphological analysis. *Arabic Text Classification : Text Preprocessing, Term Weighting, and Morphological Analysis*.

SAÂDANE H. (2015). *Le traitement automatique de l'arabe dialectalisé : aspects méthodologiques et algorithmiques*. PhD thesis, Université Grenoble Alpes.

SAADANE H. & HABASH N. (2015). A conventional orthography for algerian arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, p. 69–79.

SAIF H., ORTEGA F. J., FERNÁNDEZ M. & CANTADOR I. (2016). Sentiment analysis in social streams. *Emotions and Personality in Personalized Services : Models, Evaluation and Applications*, p. 119.

SAIF M. MOHAMMAD, MOHAMMAD SALAMEH S. K. (2016). Sentiment lexicons for arabic social media. In *Proceedings of 10th edition of the the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

SALAMA R. A., YOUSSEF A. & FAHMY A. (2018). Morphological word embedding for arabic. *Procedia computer science*, **142**, 83–93.

SALAMEH M., MOHAMMAD S. & KIRITCHENKO S. (2015). Sentiment after translation : A case-study on arabic social media posts. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics : Human language technologies*, p. 767–777.

SALIB M. B. (1981). *Spoken Arabic of Cairo*. American University in Cairo Press.

SANTOS C. D. & ZADROZNY B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, p. 1818–1826.

SAUNDERS C. H., PETERSEN C. L., DURAND M.-A., BAGLEY P. J. & ELWYN G. (2018). Bring on the machines : Could machine learning improve the quality of patient education materials ? a systematic search and rapid review. *JCO clinical cancer informatics*, **2**, 1–16.

SAXENA D., GUPTA S., JOSEPH J. & MEHRA R. (2019). Sentiment analysis. *Journal Homepage : http ://www. ijesm. co. in*, **8**(3).

SCHMITT M., STEINHEBER S., SCHREIBER K. & ROTH B. (2018). Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 1109–1114.

SCHNABEL T., LABUTOV I., MIMNO D. & JOACHIMS T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 298–307.

SCHUSTER M. & PALIWAL K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, **45**(11), 2673–2681.

SFAR I. (2006). Fonctions syntagmatiques et incorporation dérivationnelle (affixale et par schèmes) des noms de professions.

SHAPIRO P. & DUH K. (2018). Morphological word embeddings for Arabic neural machine translation in low-resource settings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, p. 1–11, New Orleans : Association for Computational Linguistics.

SHBOUL B. A., AL-AYYOUB M. & JARARWEH Y. (2015). Multi-way sentiment classification of arabic reviews. *the 6th International Conference on Information and Communication Systems (ICICS 2015)*.

SHOUKRY A. & RAFEA A. (2015). A hybrid approach for sentiment classification of egyptian dialect tweets. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, p. 78–85 : IEEE.

SINGH T. & KUMARI M. (2016). Role of text pre-processing in twitter sentiment analysis. *Procedia Computer Science*, **89**(Supplement C), 549–554.

SOLIMAN A. B., EISSA K. & EL-BELTAGY S. R. (2017). Aravec : A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, **117**, 256–265.

SOLIMAN T. H., ELMASRY M., HEDAR A. & DOSS M. (2014). Sentiment analysis of arabic slang comments on facebook. *International Journal of Computers & Technology*, **12**(5), 3470–3478.

SOLÍS-AVILÉS E., ESPINOZA A. H., ORTIZ-ZAMBRANO J. & VARELA-TAPIA E. (2018). Sentiment analysis in education domain : A systematic literature review. In *Technologies and Innovation : 4th International Conference, CITI 2018, Guayaquil, Ecuador, November 6-9, 2018, Proceedings*, volume 883, p. 285 : Springer.

SONG M. (2019). Health social network analytics : Analysis of chronic diseases with extracted entities and their relations. *J Med Internet Res*, **21**(6), e12876.

SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I. & SALAKHUTDINOV R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, **15**(1), 1929–1958.

STEMMER G., STEIDL S., NÖTH E., NIEMANN H. & BATLINER A. (2002). Comparison and combination of confidence measures. In *International Conference on Text, Speech and Dialogue*, p. 181–188 : Springer.

STONE C. J. (1984). Classification and regression trees. *Wadsworth International Group*, **8**, 452–456.

SUTSKEVER I., MARTENS J., DAHL G. & HINTON G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, p. 1139–1147.

TABOADA M., BROOKE J., TOFILOSKI M., VOLL K. & STEDE M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics*, **37**(2), 267–307.

TAHA H. Y. (2013). Reading and spelling in arabic : Linguistic and orthographic complexity. *Theory & Practice in Language Studies*, **3**(5).

TAI K. S., SOCHER R. & MANNING C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv :1503.00075*.

TANG D., QIN B. & LIU T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, p. 1422–1432.

TANG D., WEI F., QIN B., YANG N., LIU T. & ZHOU M. (2016). Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, **28**(2), 496–509.

TANG D., WEI F., YANG N., ZHOU M., LIU T. & QIN B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, p. 1555–1565.

TAYLI M. & AL-SALAMAH A. I. (1990). Building bilingual microcomputer systems.(integrated arabic system). *Communications of the ACM May*, **33**(5), 10.

TONG R. M. (2001). An operational system for detecting and tracking opinions in on-line discussion. In *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*, volume 1.

TOUAHRI I. & MAZROUI A. (2019). Studying the effect of characteristic vector alteration on arabic sentiment classification. *Journal of King Saud University-Computer and Information Sciences*.

TURIAN J., RATINOV L. & BENGIO Y. (2010). Word representations : a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, p. 384–394 : Association for Computational Linguistics.

TURNEY P. D. (2002). Thumbs up or thumbs down ? : semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, p. 417–424 : Association for Computational Linguistics.

TURNEY P. D. & LITTMAN M. L. (2003). Measuring praise and criticism : Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, **21**(4), 315–346.

VAPNIK V. (1982). Estimation of dependences based on empirical data [in russian] 1979. *English Translation by Kotz, Samuel in*.

VERMA B. & THAKUR R. S. (2018). Sentiment analysis using lexicon and machine learning-based approaches : A survey. In *Proceedings of international conference on recent advancement on computer and communication*, p. 441–447 : Springer.

WEGRZYN-WOLSKA K., BOUGUEROUA L., YU H. & ZHONG J. (2016). Explore the effects of emoticons on twitter sentiment analysis. *Comput. Sci. Inf. Technol.*, **2**, 65.

WIDROW B. & HOFF M. E. (1960). *Adaptive switching circuits*. Rapport interne, Stanford Univ Ca Stanford Electronics Labs.

WILLIAMS R. J. & ZIPSER D. (1995). Gradient-based learning algorithms for recurrent. *Backpropagation : Theory, architectures, and applications*, **433**.

WILSON T., WIEBE J. & HOFFMANN P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

XIA Y., CAMBRIA E., HUSSAIN A. & ZHAO H. (2015). Word polarity disambiguation using bayesian model and opinion-level features. *Cognitive Computation*, **7**(3), 369–380.

XU L., REN J. S., LIU C. & JIA J. (2014). Deep convolutional neural network for image deconvolution. In *Advances in neural information processing systems*, p. 1790–1798.

YOUSSEF M. & EL-BELTAGY S. R. (2018). Moarlex : An arabic sentiment lexicon built through automatic lexicon expansion. *Procedia computer science*, **142**, 94–103.

ZEYER A., ALKHOULI T. & NEY H. (2018). RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. *CoRR*, **abs/1805.05225**.

ZHANG S., CHOROMANSKA A. E. & LECUN Y. (2015). Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, p. 685–693.

ZHOU P., QI Z., ZHENG S., XU J., BAO H. & XU B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv :1611.06639*.

ZITOUNI I., SORENSEN J. S. & SARIKAYA R. (2006). Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, p. 577–584 : Association for Computational Linguistics.

ZOUAGHI A., ZRIGUI M. & AHMED M. B. (2004). Une structure sémantique pour l'interprétation des énoncés en langue arabe. *JEP-TALN*.

ZRIBI I., BOUJELBANE R., MASMOUDI A., ELLOUZE M., BELGUITH L. H. & HABASH N. (2014). A conventional orthography for tunisian arabic. In *LREC*, p. 2355–2361.