Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Translittération et normalisation de la langue arabe pour l'analyse de sentiments dans les médias sociaux

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Ingénierie Multilingue

par

Lilia FEREDJ

Directeur de mémoire :

Damien Nouvel

Encadrant :

Pedro Miguel Dias Cardoso

Année universitaire 2015/2016

Remerciements

En guise de reconnaissance, je tiens à témoigner mes sincères remerciements à toutes les personnes qui ont contribué de près ou de loin au bon déroulement de mon stage de fin d'étude et à l'élaboration de ce modeste travail.

Mes sincères gratitudes à mon directeur de mémoire M. Damien NOUVEL pour la qualité de son enseignement, ses conseils et son intérêt incontestable qu'il porte à ce mémoire, je le remercie aussi pour son attention et sa patience.

Je tiens à remercier mon tuteur de stage M. Pedro CARDOSO ainsi toute l'équipe de Data Science chez Synthesio pour leur patience, conseils, orientation et l'intérêt qu'ils ont porté à mes réalisations.

Enfin, je n'oserais oublier de remercier tout le corps professoral du Master Ingénierie Multilingue de L'INALCO, pour le travail énorme qu'il effectue pour nous créer les conditions les plus favorables pour le déroulement de nos études.

RÉSUMÉ

Ce travail aborde le sujet de la translittération et la lemmatisation de la langue arabe pour l'analyse des sentiments des messages issues du web social. Nous avons développé un système de translittération à base de règles et contribué à l'amélioration du système de lemmatisation. Les règles morphologiques et grammaticales ont été implémentées sous forme d'une chaîne de traitement. Pour la translittération on s'est intéressé à l'arabizi, un arabe dialectal écrit en lettres latines, vers l'arabe en caractères standards. Pour pouvoir transcrire l'arabizi, on a besoin d'abord de l'identifier,ceci étant réalisé par un algorithme d'apprentissage automatique, à l'aide de l'outil Keras. La translittération et la lemmatisation sont des structures interconnectées qui sont exploitées pour le but final du projet, la détection des sentiments dans les documents issues de médias sociaux. L'approche de l'analyse des sentiments est réalisée par une méthode hybride reposant sur un lexique et de l'apprentissage automatique. La mesure d'évaluation des systèmes de translittération et de lemmatisation a été effectué avec la distance de Levenshtein. La performance du système est améliorée après chaque évaluation en définissant des règles plus précises et plus puissantes. Après toutes les expérimentations, nous avons atteint une de F-mesure de 75,23% pour la translitération, 93% pour la lemmatisation et 92% pour la détection des sentiments.

Mots clés : *translittération, langue arabe, arabizi, analyse des sentiments, distance de Levenshtein, apprentissage automatique.*

TABLE DES MATIÈRES

Remerciements **3**

Résumé **5**

Liste des figures **8**

Liste des tableaux **8**

Introduction **9**

I Contexte général **11**

1 Présentation de l'entreprise **13**
1.1 A propos de Synthesio . . . 13
1.2 Couverture des données . . . 14
1.3 Outils en TAL . . . 15

2 Concepts et définitions autour du contexte **17**
2.1 La langue arabe . . . 17
2.2 Arabe dialectal . . . 18
2.3 Arabizi . . . 18
2.4 Translittération . . . 18
2.5 Lemmatisation VS racinisation . . . 19
2.6 Langue agglutinante . . . 19
2.7 Voyelles brèves . . . 19
2.8 Voyelles longues . . . 20
2.9 Analyse de sentiment . . . 20
2.10 Polarité . . . 21

3 État de l'art **23**
3.1 Travaux connexes . . . 23

4 Formulation du problème et objectifs **27**

II Méthodes et Expérimentations **29**

5 Corpus et outils **31**
5.1 Collection et présentation des données . . . 31
5.2 Annotations . . . 32

6 Translittération **35**
6.1 Prétraitement . . . 35
6.2 Algorithme . . . 35
6.3 Évaluation . . . 37
6.4 Discussions . . . 38

7 lemmatisation **41**

7.1 Méthodes 41
7.2 Résultats 42
7.3 Discussions 43

8 Analyse de sentiments **45**
8.1 Méthodes 45
8.2 Résultats 47
8.3 Discussions 47

Conclusion générale **51**

Bibliographie **53**

A Annexes **55**
A.1 Exemple de code d'une règle de translittération 55
A.2 Code de quelques règles de racinisation ajoutées 56
A.3 Script d'évaluation avec la distance de Levenshtein 56

Liste des figures

1.1 Distibution des verbatims des trois derniers mois sur les différentes thématiques 14
1.2 Nombre de verbatims par pays 15
1.3 Le word cloud 16
5.1 Annotation avec **BRAT** 32
5.2 Exemple d'annotation des sentiments avec **Brat** 34

Liste des tableaux

5.1 Distribution des données par dialecte 31
5.2 Distribution de la totalité des données pour l'analyse de sentiment 32
6.1 Résultats de l'évaluation de la translittération 38
6.2 Extrait du dictionnaire des mots en longueur de 3 39
6.3 Résultats d'amélioration avec les dictionnaires 39
7.1 Évaluation avant toutes modification dans le système de racinisation 42
7.2 Évaluation après les modifications dans le système de racinisation 42
7.3 Évaluation du système de racinisation 43
8.1 Création du lexique sémantique 45
8.2 Résultats de détection de sentiment après la 5ème correction 47
8.3 Résultats de détection de sentiment après la 10ème correction 47
8.4 Correction de polarité des mots 48
8.5 Résultats de détection de sentiment après le sampling 48

INTRODUCTION

Au cours des dernières décennies du 20ème siècle et surtout depuis les années 1990, les technologies de communication conçues pour traiter des textes occidentaux sont devenues de plus en plus répandues dans le monde arabe, tels que les ordinateurs personnels, le World Wide Web, le courrier électronique et les SMS. La plupart de ces technologies avait à l'origine la possibilité de communiquer en utilisant l'alphabet latin uniquement, et certains d'entre eux ne disposent toujours pas de l'alphabet arabe en option. Par exemple, *"Internet Explorer 5.0, qui a été publié en Mars 1999, a été la première version du navigateur supportant l'affichage de l'arabe. Windows Mobile et Android ne prennent pas en charge l'arabe sauf par le soutien d'un tiers, jusqu'à ce que les versions 6.5x et 3.x respectivement soient publiées"* [1]. En conséquence, les utilisateurs arabophones communiquaient avec ces technologies, en arabe, mais en utilisant des caractères latins, c'est à dire en faisant une translittération de leurs messages. Ce système d'écriture est souvent appelé arabizi, arabish, franco-arabe.. Pour gérer ces lettres arabes qui ne disposent pas d'un équivalent phonétique approximative dans l'alphabet latin, chiffres et autres caractères ont été affectés. Par exemple, le chiffre "3" est utilisé pour représenter la lettre arabe "ع", car il existe une similitude visuelle entre la lettre arabe et son remplacement numérique. Malgré le développement de différentes plate-formes supportant l'arabe, l'arabizi continue à être populaire en raison de la familiarité des utilisateurs avec elle ainsi que les facilités pour les utilisateurs d'utiliser un clavier anglais par rapport à un clavier arabe. L'arabizi est utilisé pour écrire à la fois l'arabe standard (MSA) ainsi que différents dialectes arabes (qui manquent la plupart du temps de conventions orthographiques et peuvent différer beaucoup, morphologiquement et phonétiquement, du MSA). En outre, en raison du fait que beaucoup d'arabophones sont bilingues (leur seconde langue étant souvent l'anglais ou le français), un autre phénomène couramment observé est la présence de l'anglais (ou français) et l'arabizi mélangés dans les phrases, puisque les utilisateurs basculent entre les deux langues.

Dans ce travail, nous allons nous concentrer sur la tâche de l'identification de portions de textes en arabizi et leur conversion (translittération) vers l'arabe. D'autres tâches sont aussi liées à la translittération et ont un rôle primordial pour le but du projet qui sont la lemmatisation et l'analyse des sentiments. Dans la première partie de ce mémoire, nous allons étudier le contexte général du thème à travers de la présentation du milieu de déroulement du stage, des concepts et définitions autour de la langue arabe, un état de l'art sur la translittération en troisième chapitre, puis une étude du problème et objectifs. Ensuite dans la deuxième partie, nous présentons l'ensemble des méthodes et expérimentations pour les tâches de translittération, de lemmatisation et d'analyse de sentiment en présentant les résultats et en les discutant.

1. https ://en.wikipedia.org/wiki/Internet_Explorer

Première partie

Contexte général

CHAPITRE 1

PRÉSENTATION DE L'ENTREPRISE

Sommaire

1.1 A propos de Synthesio **13**
Produit et avantages 13
1.2 Couverture des données **14**
1.3 Outils en TAL **15**
ASA 15
Demographics 15
Word cloud 15

1.1 A propos de Synthesio

Synthesio est une entreprise de l'industrie de l'écoute dans des médias sociaux, fondée en 2006 à Paris par Thibault Hanin et Loic Moisand. Elle fournit des solutions de surveillance et d'analyse évolutive à des marques et agences dans le monde entier. Elle aide ses clients à écouter, comprendre et collaborer avec leurs consommateur. Synthesio aide les clients à filtrer le bruit dans les médias sociaux pour, se concentrer sur les conversations qui comptent le plus pour leur entreprise, suivre et mesurer l'e-réputation de leurs produits, collaborer, engager les clients en temps réel et gérer les relations de consommation sur une plate-forme. En fournissant des données de médias sociaux de plus de 190 pays et 80 langues, les clients reçoivent des indications sur les opinions de leur marché, les désirs et les besoins, pour optimiser la valeur commerciale de leurs produits.
Synthesio est une équipe d'innovateurs partageant la même passion pour les technologies sociales. Avec plus de 150 collaborateurs et 4 bureaux autour du globe : Paris, Londre, New York et Singapour.

Synthesio permet de suivre et d'analyser les conversations de plus de 100 000 sites internet répartis. Peu importe l'endroit où les gens parlent d'une marque, Synthesio trouve et écoute les conversations qui intéressent ses clients. Grâce à son analyse automatique du sentiment disponible en 21 langues, Synthesio permet en particulier de couvrir l'arabe et d'écouter WeChat.

Produit et avantages

Synthesio propose en particulier les services suivants :
— Capacité à relier les données sociales au objectifs business de l'entreprise.
— Ambition de permettre aux clients d'atteindre un niveau de maturité avancé dans leurs initiatives d'écoute.
— Écosystème solide de partenariats et l'ouverture de son API
— Requêtes dans plusieurs langues pour réaliser une veille mondiale
— Filtrage par langue, région ou par pays sur l'ensemble du tableau de bord ou sur chaque widget
— Traduction des mentions à l'aide d'un bouton de traduction Google intégré.

1.2 Couverture des données

L'équipe dédiée au sourcing chez synthesio a la responsabilité d'ajouter des nouveaux sites au corpus Synthesio. La plupart des fournisseurs mettent en avant le nombre de sources qu'il écoutent. L'équipe sourcing a à ce jour collecté plus de 36 milliards de mentions (une mention pourrait être un document ou un message de facebook, twitter ou un post d'un forum) dans leur base de données globale et en intégrant chaque jour 30 millions supplémentaires depuis 100,000 sources. Voyons ici l'état de la base de donnée au 02 Août 2016 :

- sources actives : 2,291,881
- nombre total d'articles : 58,798,309,531
- articles du derniers mois : 1,189,619,253
- articles de la dernières semaine : 289,959,984
- articles du jour : 28,756,288

Les données de Synthesio sont catégorisées en différentes thématiques, cela grâce à un système d'identification automatiques des tendances appelé *Affinities detection* qui détecte les tendances du marché à l'aide de l'Insight Tree (un logiciel qui aide à construire des arbres de décision afin d'optimiser et de documenter les processus de prise de décision et de modéliser des cas commerciaux complexes). Voici quelques unes de ces thématiques : IT & Technology, Sport, Love & Familly... La figure 1.1 indique la répartition de 97 923 317 verbatims [1] sur les trois derniers mois.

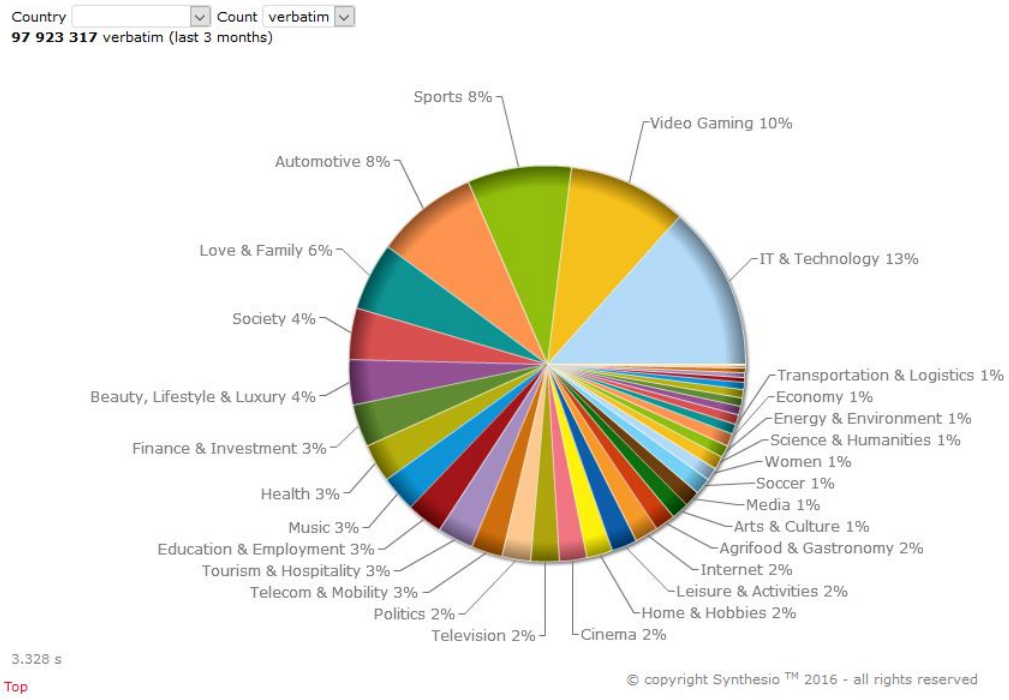

FIGURE 1.1 – Distibution des verbatims des trois derniers mois sur les différentes thématiques

1. Un verbatim est la même chose qu'une mention, deux appellations différentes employées chez Synthesio

Voici dans la figure 1.2 le nombre de verbatims pour chaque type de média (twitter, forums..) sur des zones géographiques spécifiques comme la France, l'Angleterre, l'Espagne, etc.

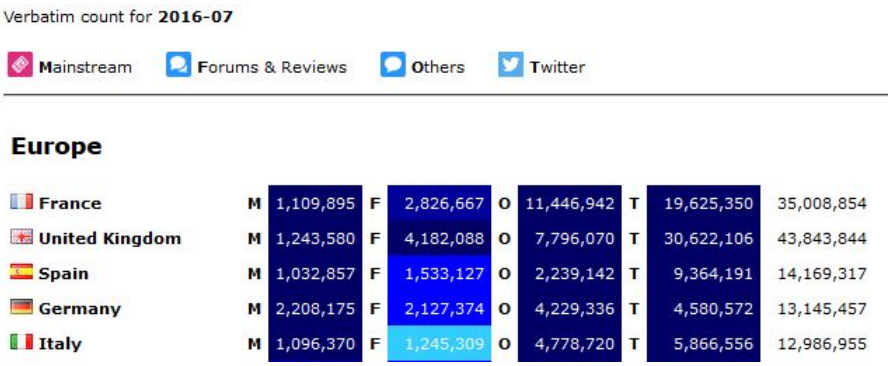

FIGURE 1.2 – Nombre de verbatims par pays

Synthesio possède un système de suppression automatique du bruit basé sur l'analyse humaine ainsi que la détection de spams et de doublons pour garantir une qualité optimale de ses données.

1.3 Outils en TAL

ASA

Synthesio possède des algorithmes de traitement automatique de langues pour la détection des tendances mais elle possède aussi un système d'analyse automatique du sentiment (ASA) qui permet de détecter les sentiments liés aux marques et d'analyser leur réputation. Le système identifie le ton d'une conversation et la catégorise comme positive, négative ou neutre. Cela permet d'extraire des opinions parmi les conversations des clients. Synthesio offre un taux de pertinence qui varie entre 65 et 70% avant toute optimisation dédiée. Avec les équipes de Customer Sucess, ce taux peut atteindre jusqu'à 80%. On verra plus plus en détails le fonctionnement de l'ASA dans le chapitre 8.

Demographics

Ce module se focalise sur la détection des tendances, *Demographics* détecte les l'âge, le sexe, statut matrimonial et la situation familiale des utilisateur sur les réseaux sociaux. Cela permet aux clients de filtrer les données selon les catégories sociales des auteurs des mentions. Par exemple selon le sexe d'un utilisateur (*male, female*)ou la statut matrimonial (*single, married...*)

Word cloud

Pour créer les nuages de mots, les calculs sont fait en se basant sur la fréquence des mots dans un ensemble de données. Ils ne sont pas calculés à partir du nombre de mentions mais de la fréquence des mots dans chaque mention. Cela qualifie les mots qui sont plus fréquents dans peu de mentions par rapport à ceux qui le sont mais dans plusieurs mentions. Par exemple : soit un mot m1 présent 2 fois dans 20 mentions (40 fois) et un mots m2 présent 5 fois dans 10 mentions (50 fois), alors le mots m2 est plus fréquent que m1 et avec une forme plus grande dans le word cloud. Les mots dans le Word Cloud sont colorés pour représenter le sentiment des mentions dans lesquelles ils se trouvent (vert : positif, rouge : négatif, blanc : neutre et mauve : non assigné). Le Word Cloud supporte 36 langues.

FIGURE 1.3 – Le word cloud

Remarque

Toutes les données et informations citées dans cette partie [Présentation de l'entreprise] ont été récupérées des sources appartenant à Synthesio (site, manuels [Synthesio, 2014b], [Synthesio, 2014a],...etc.)

CHAPITRE 2

CONCEPTS ET DÉFINITIONS AUTOUR DU CONTEXTE

Sommaire

2.1 La langue arabe **17**
2.2 Arabe dialectal **18**
2.3 Arabizi **18**
2.4 Translittération **18**
2.5 Lemmatisation VS racinisation **19**
2.6 Langue agglutinante **19**
2.7 Voyelles brèves **19**
2.8 Voyelles longues **20**
2.9 Analyse de sentiment **20**
Niveau d'analyse 20
2.10 Polarité **21**

2.1 La langue arabe

"La langue arabe "العربية" en arabe ou "Al-arabiya" est une langue sémétique originaire de la péninsule rabique depuis le VII siècle. Le nombre de locuteurs est estimé à 270 millions de personnes au sein du monde arabe et de la diaspora arabe. Elle est devenue aujourd'hui l'une des langues les plus parlées dans le monde. Elle est la langue officielle de 28 pays et de plusieurs organismes internationaux, dont l'une des six langues officielles de l'Organisation des Nations Unies" [1] [2]. La langue arabe est connue par son importante diglossie entre l'arabe littéral, qui correspond à l'arabe standard moderne surtout écrit et l'arabe dialectal, qui comprend de nombreuses variétés régionales surtout orales. Ces langues ne sont pas toutes compréhensibles entre elles.

La langue arabe comporte un nombre assez important de consonnes en la comparant à d'autres langues latines (28 en arabe littéral) et peu de voyelles (3 voyelles et 2 longueurs en littéral, souvent un peu plus en dialectal). Elle s'écrit avec l'alphabet arabe. L'arabe est une langue accusative et flexionnelle car sa grammaire fait appel à un usage des flexions internes. Les racines des mots sont très majoritairement formées de trois lettres écrites. Ces racines permettent de former des mots, soit au moyen de lettres dites serviles qui sont des lettres ajoutées et pas originelles dans le mot, soit par le redoublement des radicales, ou encore par le changement des voyelles brèves figurées par des points diacritiques sur les lettres. C'est ainsi qu'une même racine peut se dériver en différentes classes grammaticales pour la langue (des verbes, des substantifs, des adjectifs, des adverbes...). La conjugaison

1. La diaspora arabe est constituée de l'ensemble des personnes qui s'identifient comme arabes et ne vivent pas dans un des vingt-deux pays de la Ligue arabe.
2. https://fr.wikipedia.org/wiki/Arabe

arabe est simple. Elle comporte deux temps (passé et présent), les règles sont communes à tous les verbes saufs quelques verbes qui peuvent subir quelques mutations quand leur radical contient l'une de ces trois lettres (ا،و،ي). La construction des conjugaisons est généralement directe, les verbes au passé sont suffixés mais au présent ils sont préfixés, parfois suffixés aussi. La conjugaison arabe parait pauvre, mais à l'aide de petites particules ou par le changement des voyelles brèves, on détermine le passé, le présent, le subjonctif, etc., avec autant de précision que l'on veut même au niveau du pronom personnel. La syntaxe suit l'ordre fondamental verbe-sujet-objet pour les phrases verbales, sujet-verbe-objet pour les phrases nominales. Le déterminant suit le déterminé dans le groupe nominal.

"L'arabe est une langue très riche ; les Arabes se vantent, selon Ernest Renan, d'avoir 80 mots pour désigner le miel, 200 pour le serpent, 500 pour le lion, 1000 pour le chameau et l'épée, et jusqu'à 4400 pour rendre l'idée de malheur. Le vocabulaire comprend 60 000 mots et 1726 racines différentes". [3]

2.2 Arabe dialectal

L'arabe dialectal (DA) « العربية الارجة » en arabe ou al-arabiyyah ad-darijah est une interférence linguistique entre la langue arabe et les langues locales ou voisines (égyptien, algérien, tunisien...). Elles sont couramment utilisées de façon informelle. Elles possèdent une syntaxe, un vocabulaire et une prononciation spécifiques à chacune d'elles qui en font des langues différentes de l'arabe littéraire, à tel point que la communication entre deux locuteurs, un ne parlant que l'un de ces dialectes et l'autre que l'arabe littéraire est difficile, sans être impossible. Bien que le MSA soit écrit en script arabe, le DA est souvent écrit avec un alphabet latin cela à cause des influences historiques et culturelles dues principalement au colonisations et au développement des médias.

2.3 Arabizi

Ils existe plusieurs appellations : **franco-arabic**, **arabay**, **mouarab**, **arabic chat alphabet** et **arabizi**. [Tobaili, 2016] décrit Arabizi comme un portemanteau entre araby-englizi, qui veut dire arabe-anglais et dit que c'est une tendance numérique de communication en textos arabes non-standards en utilisant l'alphabet latin. Tandis que [Al-Badrashiny et al., 2014] affirment que l'arabizi est une orthographe spontanée utilisé pour écrire l'arabe dialectal en utilisant l'alphabet latin, des chiffres et d'autres symboles couramment trouvés sur divers dispositifs tels que la ponctuation. Arabizi est régulièrement utilisé par des locuteurs arabes pour écrire dans les médias sociaux et les SMS et les applications de chat. Par exemple la phrase : "**rah neb3ethelak mail**" est une phrase en dialecte arabizi qui s'écrit en arabe "راح نبعثلك ميل" et qui veut dire "je vais t'envoyer un mail".

2.4 Translittération

Selon une source anonyme sur le web *"La translittération consiste à représenter des caractères d'une écriture par les caractères d'une autre. l'opération étant réversible, l'emploi de signes diacritiques ou de digraphes permet de résoudre le problème du nombre différent de caractères entre les alphabets des deux systèmes d'écritures (arabe 28 lettres et anglais 26). Cette opération de conversion a pour objectif de permettre la reconstitution automatique et univoque de l'écriture originale. En un mot, la translittération d'un texte translittéré doit retourner l'original. On utilise pour cela des standards de normalisation. Ces standards de normalisation définissent les règles de translittération : quels caractères sont remplacés par quels autres et dans quel cas. La romanisation (ou latinisation) est la translittération d'une écriture non latine vers une écriture latine. La translittération, même si elle pourrait donner l'impression*

3. http ://www.agoravox.fr/actualites/religions/article/la-langue-arabe-son-histoire-son-77459

d'être un mot appartenant la langue dont le système d'écriture est utilisé, n'est pas une traduction puisqu'elle s'écrit dans son système alphabétique ou syllabique mais pas sémantique". [4]

2.5 Lemmatisation VS racinisation

La lemmatisation désigne l'analyse lexicale d'un texte dans le but de regrouper les mots d'une même famille. Les mots d'une même famille sont donc réduits en une unique entité appelée « *lemme* » ou « *forme canonique* ». La lemmatisation regroupe les différentes formes que peut prendre un mot par un mécanisme de flexion (comme la déclinaison ou la conjugaison) : le verbe à l'infinitif, le verbe conjugué à tous les temps, le le nom au singulier ou au pluriel, les pronoms déclinés, etc.
En arabe les changement sont ainsi :

- Les verbes sont ramenés à la 3e pers. sg. de l'accompli actif, sauf dans le cas de certains verbes figés n'ayant qu'une conjugaison partielle.
- Les noms variables sont ramenés à la forme du nominatif sg. (masc. pour les noms qualificatifs), les noms invariables à leur forme d'origine (masc. sg. pour les pronoms).
- Les articles sont ramenées à leur forme d'origine.

La racinisation (en anglais : *stemming*) désigne l'analyse sémantique des mots afin d'identifier et de regrouper les différentes formes d'un même mot autour d'une racine appelé « *stemme* » ou « *stem* ». Le stemme est identifié par la suppression du préfixe et ou du suffixe d'un mot. La racine en arabe, appelé « *Jidhr* », « الجذر », est plus souvent trilitère (composée de trois consonnes). Elle est la suite des consonnes formant le radical du mot. La racine est un élément important dans les langues sémitiques : associée à un schème.

Le schème, en arabe « الميزان » qui veut dire « *Balance* », est un moule précis sur lequel un mot est construit. Par exemple à partir de la racine : « كتب » en schème « فعل », l'on forme le nom « كاتب », « *écrivain* », selon le schème des noms de la première forme, en l'occurrence : « فاعل ». Un point différenciant important entre la lemmatisation et la racinisation : un lemme est un mot réel de la langue analysée contrairement à la racine qui n'est généralement pas un mot réel car basé sur la proximité sémantique.

2.6 Langue agglutinante

« Les langues agglutinantes forment un sous-groupe des langues flexionnelles. Le terme de *"langue agglutinante"* a été créé en 1836 par le linguiste allemand *Wilhelm von Humboldt* » [5]. Le terme est formé à partir du verbe latin *agglutinare*, signifiant «*coller ensemble*». Une langue agglutinante est une langue dans laquelle les flexions morphologiques sont formées par l'assemblage d'éléments basiques (morphèmes). Les morphèmes combinés sont alors des affixes qui peuvent être déterminés individuellement. Une langue agglutinante a un nombre important de morphèmes par mots. Par exemple prenant le mot arabe suivant : " اللاعبين " qui veut dire *"les joueurs"* se décompose en : " ال " "les" article défini + " لاعب " "joueur" qui est formé à partir de la racine" لعب " + suffixe pour former le pluriel des noms masculins " ين " qui correspond au "s".

2.7 Voyelles brèves

En arabe, les voyelles brèves sont appelées « تشكيل » « Tachkil », Elles sont quatre petits symboles qu'on met au dessous ou au-dessus des lettres pour les vocaliser. Ces diacritiques on les appelle "dhamma", "kasra", "fat'ha", "soukun" qui correspondent respectivement aux

4. http ://www.translitteration.com/qu-est-ce-que-la-translitteration/fr/
5. https ://fr.wikipedia.org/wiki/Langue_agglutinante

signes : أُ, إِ, أَ, آ. Pour la lettre "s", " س ", si on veut former des syllabes comme "so", "si", "sa", "s", on ajoute les signes diacritiques et on aura ses syllabes suivantes : سُ, سِ, سَ, سْ. Elles ne sont généralement pas écrites, sauf parfois dans les textes sacrés et didactiques, auquel cas l'on dit de ces textes qu'ils sont « vocalisés ».

2.8 Voyelles longues

En arabe, les voyelles brèves sont appelées « حروف المد ». Les voyelles longues sont des voyelles dont le son est allongé lors de la prononciation. Les voyelles longues sont formées à partir de trois lettres : ا : alif, و : waw, ي : ya. Elles sont des allongements phonétiques des voyelles courtes mais ils peuvent être utilisées comme des consonnes et porter elles aussi des voyelles brèves et pour cette raison on les appelle des semi-consonnes.

2.9 Analyse de sentiment

L'analyse de sentiment, en anglais *sentiment analysis* ou *opinion mining* est une partie de la fouille de textes qui étudie et identifie les opinions et attitudes présentes dans des données textuelles généralement sur de grandes quantités de données. Elle consiste de définir automatiquement la polarité que porte les sentiments (positif, négatif, neutre) présents dans un texte ou un ensemble de textes. Développée principalement depuis les années 2000 à la suite de la publication de grandes quantités de données provenant des réseaux sociaux, notamment celles de Twitter. Elle est spécifiquement utilisée en marketing par des entreprises pour analyser les commentaires des blogueurs sur le net pour positionner leurs marques sur le marché et avoir des connaissance poussées sur leur clients. De nombreux outils d'analyse automatique de sentiments sont développés afin de détecter rapidement les sentiments clés issus des messages d'internautes. Parmi eux, notament : Werfamous, SenticNet.

Niveau d'analyse

L'analyse de sentiment « basique » se focalise généralement sur une seule dimension (positif ou négatif). Il existen d'autres techniques visant à déterminer d'autres sentiments généraux comme l'envie, la colère, la frustration ou la joie. L'analyse peut s'effectuer à différents niveaux :

- **Au niveau de document :** détermine l'opinion générale de l'ensemble du document qui peut être un message, commentaire, tweet... cette analyse fonctionne bien pour des documents qui présentent un point de vue précis, mais moins pour des comparaisons de plusieurs sujets car elle ne fera pas la différence de sentiments pour chaque sujet abordé.
- **Au niveau de la phrase :** détermine l'opinion générale d'une phrase (positive, négative ou neutre).

Il existe plusieurs méthodes pour la détection de sentiments, la plus répandue parmi ces méthodes est *la classification par apprentissage automatique* qui fait partie de l'ensemble des méthodes supervisées. Aussi les méthodes non-supervisées comme l'analyse syntaxique et les règles de score qui attribuent pour chaque mot son score de sentiment dans l'intervalle [-1, +1]
La langue arabe est pauvre en développement d'outils d'analyse de sentiment. Il y a très peu de travaux réalisés à cet égard. *"La plupart des travaux de recherche effectués dans ce domaine ont été menés sur des langues européennes (anglais) et asiatiques (japonais et chinois). Néanmoins, très peu de travaux ont été réalisés sur le plan des langues qui sont morphologiquement riches comme l'arabe notamment quand il s'agit d'un dialecte non-standard"* comme le mentionne [Tobaili, 2016]

2.10 Polarité

La polarité est l'intensité du jugement qu'un individu porte sur un objet. Elle consiste de façon générale à déterminer si l'opinion d'un document ou d'un message sur le sujet est positive ou négative. La polarité d'un message est dite positive ou négative si l'intensité moyenne de tous ses mots l'est. Pour cela, à fin de savoir la polarité d'un message, il faut d'abord analyser individuellement la polarité de chaque mot. Cette méthode implique l'utilisation des dictionnaires qui référencent des instances de phrases déjà existantes et pour lesquelles des émotions ont déjà été identifiées.

CHAPITRE 3

ÉTAT DE L'ART

Sommaire

3.1 Travaux connexes 23

3.1 Travaux connexes

Dans cette section, nous allons étudier l'état des connaissances existantes et les travaux réalisés par des chercheurs antérieurs dans le domaine de translittération arabe en général et en particulier la conversion de l'arabe en script latin (arabizi) vers l'arabe en caractères arabe. Beaucoup de travaux ont été conduits sur le sujet ces dernières années. Ce qui démontre les besoins pour cette tâche car les utilisateurs des médias sociaux ont beaucoup plus tendance à utiliser l'arabe avec un clavier latin, qu'ils trouvent plus facile à saisir, contrairement aux caractères arabes. La translittération automatique de l'arabe dialectal est une tâche assez compliquée et laborieuse, étant donné que la langue arabe est une collection de variétés : (MSA) Modern Standard Arabic qui est l'arabe littéraire enseigné dans les écoles avec une orthographe standard et les différentes formes de l'arabe dialectal égyptien, algérien, tunisien...qui sont de plus en plus utilisés de façon informelle le web , et qui n'ont aucune norme d'orthographe. Bien que les deux (MSA et DA) sont généralement écrites avec le script arabe, l'arabe dialectal est parfois écrit avec l'alphabet latin.

Le premier travail réalisé en translittération arabe a été développé à Xerox par [Buckwalter, 1990] dans les années 1990. C'était le seul système de translittération vers l'ASCII, qui représente l'orthographe arabe strictement lettre par lettre, à la différence des systèmes de romanisation plus communs qui ajoutent des informations morphologiques des mots arabes. Ainsi, par exemple, un waw "و" est translitteré w indépendamment du fait qu'il est réalisé comme une voyelle /u :/ ou une consonne /w/. Seulement, lorsque le waw est modifié par une hamzah "ؤ" la translittération se change à &. Depuis que la table de translittération arabe de Buckwalter a été développée, plusieurs autres variantes ont vu le jour, même si elles ne sont pas toutes normalisées. La première version Buckwalter translittération n'est pas compatible avec XML, donc des versions XML safe ont été créées, modifiant les caractères suivants : "<", ">" et "&" qui représentaient les caractères suivant "أ", "إ" et "ؤ" respectivement en les translittération I, O, et W respectivement.

Pour des buts de traduction automatique des noms propres [Arababi et al., 1994] ont développé un algorithme hybride qui automatise le processus en temps réel en utilisant les réseaux de neurones et un système basé sur les connaissances de voyellisation arabe. Un réseau de neurones supervisé filtre les noms peu fiables, en passant les noms fiables dans la base des connaissances pour la romanisation. Cette approche, développée à l'IBM Federal Systems Company, est applicable à une grande variété d'objectifs, y compris le traitement des visas et le traitement des documents par les patrouilles frontalières.

[Stalls and Knight, 1998] présentent un système de translittération basé sur un modèle

génératif de phonèmes, d'un mot source anglais vers l'arabe en plusieurs étapes, chacune étant définie comme un modèle probabiliste qui représente une machine à états finis. Premièrement, le mot est généré à partir de probabilité des uni-grammes. Après le mot est converti en séquence de sons phonétiques en utilisant un dictionnaire de prononciations. Enfin, les séquences de phonèmes sont converties en un mot arabe. Ce modèle proposé par Stalls et Knight était pertinent à son époque, cependant l'utilisation de dictionnaire de prononciations pour faire la conversion ne permet de translittérer que les mots qui ont une prononciation connue dans ce dictionnaire.

[Al-Onaizan and Knight, 2002] est une continuation du travail précédent. Ils ont proposé une approche complémentaire au modèle phonétique réalisé antérieurement en 1998. Un système de translittération des noms propres basé sur le son et l'orthographe du mot a été ajouté. Le modèle phonétique lui même a été amélioré. Pour éviter la production des mots déformés, les auteurs ont ajouté des caractéristiques sur la position du phonème dans le mot (début, milieu, fin) pour y appliquer des restrictions avant de générer le mot en langue cible. Le modèle orthographique est principalement réalisé pour la translittération des noms propres en langues autres que l'anglais car l'implantation des dictionnaires pour toutes les langues est considérablement fastidieux. Ce modèle transforme directement les séquences de lettres en anglais en séquences de lettres en arabe avec une probabilité de génération P(a | w) qui est entraînée sur une liste de noms anglais/arabe. Comme la prononciation n'est pas modélisée, plusieurs paires d'autres langues ont été facilement obtenues. Par la suite le modèle est étendu pour contenir des tri-grammes de lettres et les uni-grammes de mots.

[Chalabi and Gerges, 2012] ont présenté une approche hybride qu'ils ont conçue et mise en œuvre pour construire un système de translittération arabe qui a été plus tard étendu progressivement pour couvrir d'autres scripts. Leur approche tire parti du travail accompli par [Sherif and Kondrak, 2007] et [Cherry and Suzuki, 2009] fortement inspiré de l'approche basée sur la traduction statistique automatique au niveau de la phrase conçue par [Och, 2003]. Leur mise en œuvre initiale visait la translittération de l'arabe dialectal, écrit en caractères latins, vers arabe dialectal écrit en caractères arabes. Plus tard, le même système a été utilisé pour transcrire le texte anglais écrit en caractères latins vers l'arabe écrit en caractères arabes et vice-versa. Pour la collecte des données les auteurs ont utilisé un outil de crawling nommé « Blog Muncher » en plus de la collecte manuelle et la transcription d'appels téléphoniques. Le processus de translittération est effectué en 2 phases, dans la première phase un générateur de candidats génère toutes les hypothèses possibles à partir d'un module de translittération qui est composé de plusieurs règles et leurs probabilités. Dans la deuxième phase, les hypothèses générées sont notées et classées en fonction d'une combinaison linéaire de 3 modèles différents : le modèle original de translittération, le modèle du mot de la langue cible, et le modèle au niveau de caractère.

Le travail de [Darwish, 2013] se divise en deux parties : premièrement, la détection de l'arabizi et deuxièmement, la conversion de l'arabizi vers l'arabe. Il a construit une collection de 3452 paires (arabizi-arabe) pour l'entraînement et 1385 paires pour le test. La transformation est réalisée manuellement par un arabophone en langue maternelle. Ensuite, il a appliqué la normalisation de [Habash, 2010] pour les deux corpus de l'entraînement et du test. Cette normalisation permet l'écriture simplifiée de plusieurs lettres en un seul format, par exemple les lettres (أ, إ, آ) ont été transformées en (ا)L'auteur a choisi d'enlever la répétition des mêmes lettres au dessus de 2 fois, comme par exemple pour le mot "salaaaam" devient "salaam". L'arabe étant une langue agglutinante, Kareem Darwish a décidé de concaténer les conjonctions et les prépositions au mot qui les suit et que beaucoup d'internautes écrivent séparément comme les conjonctions (w, el, fel, bel...).

Pour l'entraînement, il a aligné les paires de mots au niveau des caractères en utilisant GIZA++. L'alignement produit un graphique de séquences de lettres. Il produit les correspondances entre les séquences de lettres latines et les séquences des lettres arabes avec des probabilités des correspondances associées. Sa méthode se résume en deux étapes :

- **Production des candidats :** il a implémenté un système de translittération qui correspond à celui de [Kahki et al., 2011], pour chaque mot arabizi, il produit toutes les segmentations possibles avec leur translittération au niveau des caractères. Les séquences cibles à forte probabilité ont été filtrées en utilisant le théorème de Bayes.
- **Choisir le meilleur candidat :** Il a conçu un modèle de langage sur des tri-grammes de mots en utilisant IRSTLM language modeling toolkit [Federico et al., 2008]. L'avantage de ce modèle c'est qu'il contient à la fois MSA et le dialecte. Le but étant, à l'aide de ce modèle, trouver la translittération à probabilité maximum parmi tous les candidats générés précédemment.

[Al-Badrashiny et al., 2014] ont établi un système de translittération appelé 3ARRIB "arabise !" qui consiste à faire la conversion de l'arabe translittéré en latin vers l'arabe en script arabe en suivant la convention CODA. CODA est une orthographe conventionnés pour l'arabe dialectal dont chaque mot a une seule représentation orthographique. C'est une structure pour toutes les écritures dialectales basée sur les ressemblances entre le MSA et tous les dialectes. Elle est principalement créée pour des buts de traitement automatique des langues. Leur système, pour chaque mot en arabizi, trouve le mot correct en caractères arabes dans CODA selon son contexte dans la phrase courante. En utilisant le contexte, 3ARRIB trouve le meilleur mot en script arabe parmi tous ses candidats dans CODA. Le processus de la translittération qu'il ont adopté est ainsi. Premièrement chaque phrase dans 3ARRIB doit passer par un processus de mise en minuscule et prise en charge des phénomènes liés au langage parlé. Ils ont séparé toutes les émoticônes comme (:D, :p, etc.) et les ponctuations de tous les mots, sauf l'apostrophe (') qui est utilisée pour distinguer un son d'un autre en arabe comme les deux sons : "7" pour "ح" et "7'" pour "خ". Il ont tagué ces émoticônes pour les protéger tout au long de la chaîne de traitement de translittération. Après, ils ont remplacé toutes les lettres répétées plusieurs fois dans un mots avec deux seulement, par exemple iiiii devient ii. Ce type d'écriture est utilisé souvent pour exprimer des émotions ou mettre l'accent sur des propos. En plus du split de chaque phrase par mot, 3ARRIB enregistre les changements des offsets de chaque mot pour reconstruire le même nombre de tokens à la fin du traitement. Finalement, 3ARRIB génère toutes les translittérations possibles pour chaque mot en entrée avec un système de transducteur à états finis. [1] Ensuite, les auteurs ont conduit des tests en combinaison avec ces deux composants :

- **Analyseur morphologique :** ils ont utilisé CALIMA [Habash et al., 2012]. CALIMA est un analyseur morphologique pour l'égyptien. Pour chaque candidat généré du transducteur, CALIMA fournit toutes les analyses morphologiques possibles. Si un des candidats n'est pas reconnu par CALIMA, il est filtré. Par contre, si tous les candidats ne sont pas reconnus, dans ce cas, tous sont gardés. Car pour chaque mot d'entrée, il devrait y avoir une sortie. Les auteurs ont ajouté une étape de tokenisation pour séparer les clitiques.
- **Modèle de langage :** c'est un modèle de l'égyptien entraîné sur CODA. Il ont opté pour deux méthodes de tokenisations, par des espaces et clitique. Ce qui leur a permis d'avoir deux options de modèles de langage. Pour choisir la meilleure translittération parmi tous les candidats, ils ont combiné la probabilité de chaque candidat produit par le transducteur avec la probabilité du modèle.

[Bies et al., 2014] ont créé une nouvelle ressource pour (LDC) Linguistic Data Consortium dans le cadre du projet "DARPA BOLT program" [2]. Cette ressource est un corpus des textes parallèles de arabizi-arabe développé à partir des SMS et des messages de chat sur le web. C'est une base très intéressante qui pourrait être utilisée dans des travaux potentiels de la

1. C'est une extension des automates finis qui opère sur un mot et le transforme en un ou plusieurs mots. Cela permet l'utilisation variée d'un langage, comme l'analyse morphologique des mots et leur transformation.

2. Broad Operational Language Translation est une technologie qui a l'objectif de permettre aux anglophones de rechercher et comprendre des ressources des langues étrangères informelles contenant des messages de chat et des SMS.

translittération.

Ils ont collecté 1270 conversations, dont 66% sont en arabizi, 15% entièrement en scripte arabe et 19% un mélange entre arabizi et arabe. Ils ont opté pour le système de [Al-Badrashiny et al., 2014] pour la translittération automatique. Ensuite, ils ont annoté, corrigé et normalisé tous les messages translittérés selon la convention CODA avec un outil développé par LDC qui leur a permis de faire les trois tâches à la fois pour chaque message.Les annotateurs ont suivi des guides d'annotation et de correction sur plusieurs niveaux. Pour l'annotation il ont défini des des catégories de textes qui doivent être maintenues telles quelles et et pour lesquelles il ne doit pas y avoir de changement dans le résultat, comme la ponctuation qui évoque des sentiments (?!!,!!!), les effets sonores (hhh, haha), les langues étrangères et les noms propres. Par ailleurs, concernant la correction, pour les les cognats communs au dialecte et MSA qui s'écrivent différemment ont été corrigé en MSA (hafez : حافظ mais pas حافز pour le nom propre Hafez). Ils ont également allong les voyelles (des voyelles courtes remplacées par des longues), désambiguïsé les consones qui ont la même transcription en latin (S soit "س" ou "ص"), et corrigé la segmentation incorrecte des mots et les fautes de frappe.

La tâche de la correction est laborieuse. Il faut non seulement avoir la maîtrise des conventions CODA mais aussi des connaissances linguistiques solides sur la langue arabe. 60% des désaccords entre les annotateurs étaient dues à un mauvais suivi des directives de guide d'annotation de CODA, 23% pour un mauvais suivi du guide de ponctuation, sons, imoticônes, noms propres et les mots étrangers. Effectivement, ce travail reste le premier sur un corpus parallèle arabizi-arabe qui souligne l'interaction étroite entre la forme orthographique de ce genre d'écrits informels de l'arabe et les caractéristiques spécifiques de ses dialectes, or ce corpus est restreint uniquement à l'égyptien et pas d'autres dialectes ce qui donne une piste à des travaux complémentaires ultérieurs pour d'autres dialectes.

CHAPITRE 4

FORMULATION DU PROBLÈME ET OBJECTIFS

Un nombre très important de locuteurs arabes basculent entre l'arabe dialectal et une autre langue étrangère même dans la même phrase, notamment le français pour les pays maghrébins et l'anglais pour l'égypte et le moyen-orient. Cela à cause des influences historiques où des emprunts lexicaux. Par exemple : « *La cousine ta3 yema 3andeha wahed villa kebira* », en caractères arabe : « *La cousine* تاع يما عندها *villa* كبيرة » en français ça donne : « *La cousine de ma mère a une grande villa* ». On constate bien l'utilisation des mots français (***La cousine*** *et* ***villa***) au milieu d'une phrase en algérien. Ou bien pour l'anglais dans cette phrase exprimée en égyptien : « *boso ba2a ana ro7t* ***today*** *we kont 2a3da fe* ***room*** *ma3 7awaly 10 w sa2let kol wa7ed fena* ***introduce ur self*** », en arabe : « و كنت أعدة في ***today*** بوصو بأه أنا رحت ***room*** فينا واحد كل سألت و ١٠ حوالي مع ***introduce ur self*** », en français : « *Ecoutez ! j'y étais aujourd'hui et j'étais dans la salle les environs de 10h et elle nous a demandé de nous présenter* ».
Le difficulté ne réside pas uniquement dans le fait que les messages soient exprimés en plusieurs dialectes arabes, mais aussi dans l'existence d'autres langues étrangères au sein du même message. Puisque si on opte pour translittérer sans filtrer ces langues, on aura des mots de ces langues qui seront écrits en caractères arabes mais qui n'appartiennent pas à son lexique, et l'exemple précèdent de "*la cousine*" devient " لا كوزين ". Face à ce problème, on se demande avec quel moyen on pourra distinguer et filtrer ces expressions ?

Les dialectes arabes ont beaucoup de mots communs, dont certains se prononcent ou s'écrivent de la même façon. Néanmoins, il y a également de nombreux mots spécifiques à un seul dialecte et qu'une personne qui ne maîtrise que le marocain par exemple ne parvient pas à comprendre un mot spécifique à l'égyptien. Les mots qu'ils arrivent à comprendre sans en avoir aucune connaissance sont généralement des mots qui ne diffèrent que par un phonème ou deux. Pour le verbe « dire » par exemple, il s'écrit en arabe standard « قال », marocain, algérien ou tunisien « *gal* » ou « *9al* » mais en égyptien « *Aal* ». Ce qu'un système automatique va considérer comme 3 mots différents. Le cas inverse est encore plus problématique : lorsqu'une lettre latine est utilisée pour représenter deux lettres arabes différentes. Le cas de la lettre « q » qui a deux interprétations possibles « ق » ou « ك » dans les deux mots « *qebel* », « *qter* » : « قبل », « كثر » qui veulent dire « *avant* », « *plus* » respectivement. Dans ce cas là, il est très difficile de définir des règles pertinentes pour que notre système de translittération gère correctement ce genre d'exceptions.

Un autre concept d'une langue non-standard comme l'arabizi, lorsqu des utilisateurs s'expriment à l'écrit sans suivre aucune règle grammaticale ou orthographique. En conséquence, beaucoup d'écritures peuvent être imaginées pour un seul même mot. Toutes les écritures : « ***sahebi, sahbi, sa7ebi, sa7bi, sahbbi*** » sont pour la même écriture du mot « صاحبي » qui veut dire « ***mon ami*** ». Ce qui rend la tâche encore plus difficile pour un système de traitement automatique des langues car ce genre de pratique donne un grand nombre de variété

pour écrire pour les mots des dialectes considérés. [S.Ahmed et al., 2013] mentionnent que l'emploi du dialecte et l'arabizi n'a pas encore été abordé dans littérature existante. Dès lors, beaucoup de chercheurs travaillant pour la langue arabe excluent l'arabizi de leur domaine de recherche pour la seule raison de l'indisponibilité des ressources pour ce système d'écriture comme les lexiques de mots, les lemmatiseurs et les POS taggers.
[Bies et al., 2014] affirment que l'utilisation de l'Arabizi pose de nombreux problèmes de défi pour les data scientists. Ce genre de texte represente un défi crucial pour les raisons suivantes : il est écrit en latin, il n'a aucune norme d'écriture, mélangé avec d'autres langues dans une même phrase.

Dans ce travail, on va proposer un système de translittération à base de caractères qui transforme des messages arabes écrits en caractères latin vers une écriture en caractères arabes, peu importe qu'il soit dialectal ou standard. Ce système est développé en python et incorpore plusieurs règles qui traitent un seul mot à la fois. L'analyse de chaque mot est faite en trois phases : préfixe, racine, suffixe. Ce système pourra être utilisé en amont de chaînes de traitement existantes en TAL ne prennent pas en charge l'arabizi. Autrement dit, le but de la création de cet outil est de pouvoir faire d'autres traitement ultérieurs sur le texte produit, la lemmatisation et l'analyse de sentiment sur de la langue arabe, y compris lorsque celle-ci a été écrite avec des caractères latins. Il s'agit d'uniformiser les énoncés arabes, afin qu'ils soient tous écrits avec des caractères arabes et qu'ils puissent ainsi être pris en charge par les modules dédiés à l'arabe.

Deuxième partie

Méthodes et Expérimentations

CHAPITRE 5

CORPUS ET OUTILS

Sommaire

5.1 Collection et présentation des données **31**
Corpus de translittération 31
Corpus analyse de sentiment 31
5.2 Annotations **32**
Annotation pour l'identification de l'Arabizi 32
Annotation des sentiments 33

5.1 Collection et présentation des données

Corpus de translittération

Pour l'identification de l'Arabizi et la translittération, on a constitué un corpus issu de plusieurs sources sur le Web avec des outils réservés à Synthesio. Ces sources représentent des commentaires et des conversations dans des réseaux sociaux et des forums de discussion. Ce corpus contient 1835 documents ; chaque document correspond à un commentaire ou un post, ce qui nous donne 12566 phrases et 272688 mots. Au sein du corpus les dialectes sont largement majoritaires (65% des données), dont le marocain (40%), l'égyptien (25%), le tunisien (17%), et l'algérien (13%), le reste étant constitué de dialectes du moyen-orient (5%).

Dialecte	**Nombre de documents**
Marocain	476 d
Egyptien	298 d
Tunisien	202 d
Algérien	154 d
Autres dialectes	62 d

TABLE 5.1 – Distribution des données par dialecte

Il y a 777 documents qui contiennent de l'arabe translittéré écrit en latin seulement.Nous voyons donc qu'il y a de très nombreux messages dialectaux qui mélangent l'arabe et le latin. En dehors du dialecte, le corpus comporte 20% en MSA et 15% entre langues étrangères et bruit.

Corpus analyse de sentiment

Pour cette tâche, de plus du corpus de translittération qu'on vient de décrire, on a ajouté une deuxième partie à ce corpus un ensemble de documents en arabe littéraire. Elle contient

685 documents, ce qui nous donne un total de 2520 documents pour cette tâche. Voici statistiques pour la deuxième partie : 4690 phrases et 54685 mots. Environs 80% des données sont en arabe standard et 20% en dialecte.

Pour représenter toutes les données en nombre de documents par MSA et dialecte ça donne :

Total	**2520**
Dialecte	1329
MSA	915
Autres langues & bruit	276

TABLE 5.2 – Distribution de la totalité des données pour l'analyse de sentiment

5.2 Annotations

L'annotation a été faite avec l'outil BRAT [1]. C'est un outil d'annotation rapide de corpus par interface web. Il s'agit d'ajouter des annotations aux documents textuels existants. BRAT est conçu en particulier pour l'annotation structurée, où les catégories d'annotations ne sont pas du texte libre, mais ont un format prédéfini qui peut être automatiquement traitées et interprétées par une machine.

Annotation pour l'identification de l'Arabizi

L'identification du texte translittéré est la première étape préliminaire dans le processus de translittération. Beaucoup d'utilisateurs arabes utilisent plusieurs langues à la fois et même dans la même phrase. Pour cela, une annotation est indispensable pour distinguer les segments écrits dans chaque langue. Les messages des réseaux sociaux contiennent beaucoup de données extra-linguistiques comme les hashtags, les mentions, et les URLS de référence. Mais aussi, de bruit provenant du crawling qui est toujours associé à ce type de données comme du code HTML et la redondance de contenus, notamment par les mécanismes de partage dans les médias sociaux. Pour avoir des données qualitatives et écarter le bruit, nous avons procédé à une annotation qui a été définie en 6 catégories :

- Trans : arabe écrit en latin.
- Arab : arabe écrit en caractères arabes.
- Foreign : toutes autres langues étrangères.
- Noise_foreign : autres langues mais pas de sens.
- Noise_web : balises, code html...
- Noise_other : le reste du bruit.

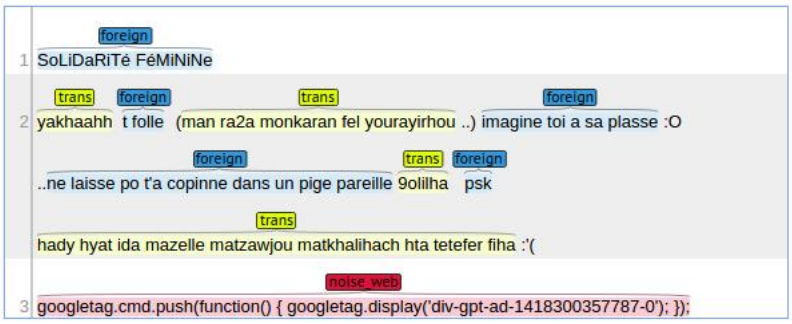

FIGURE 5.1 – Annotation avec **BRAT**

1. http ://brat.nlplab.org/

Cette méthode d'annotation discriminante a pour objectif d'alimenter un processus d'un système d'apprentissage automatique pour la détection de l'arabe translittéré. Synthesio utilise l'algorithme **keras** pour cette tâche.

Keras

Keras [2] est une bibliothèque des réseaux neurones récursifs (RNN) pour le deep learning écrite en Python. Elle a été développée en se focalisant sur l'expérimentation rapide car, être en mesure d'entraîner le plus rapidement possible est la clé pour faire une bonne recherche. Synthesio a utilisé le module *Keras layers* [3]. C'est un système dynamique de réseaux de neurones récurrents constitué d'unités neurones interconnectés interagissant non-linéairement. Elles sont reliées par des arcs qui possèdent un poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées.
Si c_1 et c_2 sont la représentation des vecteurs n-dimension de noeuds, leur parent sera également un vecteur de n-dimension. Cette architecture, avec quelques améliorations, a été utilisée avec succès dans l'analyse syntaxique des phrases en langage naturel. La complexité de cette étude augmente très rapidement avec l'augmentation le nombre de neurones.

Annotation des sentiments

Cette annotation contient 3 catégories principales : *Positive*, *Negaive*, *Neutral* ; principalement, on annote que le *Positive*, et le *Negative*. Tout le reste est *Neutral*. Cette annotation porte sur une ou plusieurs phrases et même parfois des expressions mais pas un seul mot. Dans chaque entité il y a le bloc qui définit sa polarité, *pos_bloc* pour Positive et *neg_bloc* pour Negative. C'est l'unité lexicale minimale sur laquelle on peut décider la polarité d'un message. Examinons l'exemple suivant :

Il veut dire : « Le médicament que j'ai acheté m'a beaucoup aidé, mais en vérité la séparation et très difficile ». "*nfa3ni, m'a beaucoup aidé*" est le pos_bloc du message positif en couleur bleue, "*fera9, séparation*", "*s3ib bzzzzzzzzf, très difficile*" sont les neg_bloc du message négatif en couleur rouge.

Il y a aussi une autre entité qui est une sous catégorie de pos_bloc et de neg_bloc appelée *adv_neg* qui est l'adverbe de négation qui inverse la polarité d'un message. Par exemple dans le tweet suivant : « *@ikramnk mafi shi mosta7il :)#barca* » qui veut dire "*@ikramnk ce n'est pas de l'impossible :) #barca*" est un message positif. Impossible est négatif, mais la présence d'un adverbe de négation inverse la polarité, qui devient positive. On résume ce qu'on vient de dire dans ces points :

- Positive
  - pos_bloc
    - adv_neg
- Negative
  - neg_bloc
    - adv_neg
- Neutral (le reste)

2. https ://keras.io
3. https ://keras.io/layers/about-keras-layers/

FIGURE 5.2 – Exemple d'annotation des sentiments avec **Brat**

Quelques problèmes majeures rencontrés au cours de l'annotation et le traitement de texte étaient principalement en relation avec la langue arabe, le système d'écriture de droite à gauche contrairement au latin surtout lorsque les deux sont mélangés dans la même phrase. Elle devient désordonnée et porte aucun sens et même parfois l'ordre des lettres est inversé au sein d'un mot. Alors que ces aspects sont très essentiels pour l'annotation manuelle avec l'outil Brat pour la délimitation des off-sets des unités et compréhension des messages pour l'annotation des sentiments.

CHAPITRE 6

TRANSLITTÉRATION

Sommaire

6.1 Prétraitement **35**
6.2 Algorithme **35**
Règles de translittération 35
6.3 Évaluation **37**
6.4 Discussions **38**
Deuxième évaluation 39

6.1 Prétraitement

Pour avoir des données propres, j'ai constitué un script python qui fait un nettoyage. J'ai enlevé toutes les ponctuations y compris le hashtag '#' et l'arobase '@' sauf l'apostrophe (') parce qu'elle est utilisée pour exprimer certains diacritiques en arabe, ["3" : "ع"] et [" 3' " : "غ"]. Les chiffres sont enlevés uniquement si ils sont présents seuls comme unité (dans ce cas ils expriment un chiffre) et pas dans une série de lettres qui pourrait être un mot translittéré, pour cela j'ai utilisé la fonction python `isnumeric()` qui renvoie `True` si le token est composé que des chiffres et `False` s'il y a au moins une lettre. Enfin, change la casse en minuscule et les chaînes sont segmentées par mots pour pouvoir les traiter individuellement après dans le processus de translittération.

6.2 Algorithme

Règles de translittération

Les règles sont définies par ordre de priorité qu'il faut respecter.

1. **Conversion de "و", "w" :**
   Description :
   Certains utilisateurs utilise la lettre "o" pour exprimer la lettre "w", "و", qui est utilisée comme une conjonction de coordination "et" en français. Alors que la façon la plus usuelle est "w". Comme le "o" est une voyelle brève en arabe, nous ne la convertissons pas sauf si elle est seule ou au début d'un mot pour exprimer la lettre "أ".

   Exemple :
   "chrebet kass **o** nos", "j'ai bu un verre **et** demi"

2. **Conversion de l'article défini "ال" au début des mots :**
   Description :
   L'article défini arabe, "Al", "ال" son équivalent « le » ou « la » en français, se prononce

de deux façons selon la lettre qui le suit. Les lettres arabes sont divisées en 2 groupes, "lettres solaires" et "lettres lunaires" [1]. Si la lettre qui suit le "al" est une lettre solaire, il subit une assimilation régressive et le son /l/ devient caché et on prononce que le /a/. Mais si la lettre qui le suit est une lettre lunaire, elle se prononce normalement. Ces noms viennent du fait que le mot « le soleil », "الشمس" prononcé "Ash-shams", le /l/ est assimilé puisque la lettre "ش" translittérée "sh" est une lettre solaire. Le cas contraire dans « la lune », "القمر" prononcé "al-qamar" car la lettre "ق" translittérée "q" est lunaire. Le résultat de cette assimilation du /l/ affecte même son écriture en Arabizi et certains utilisateurs ont cessé de l'écrire parce qu'il n'est pas prononcé. Donc, ils écrivent le "A" et 2 fois la lettre suivante.

Exemple :
Assimilation regressive du /l/ :
"Arraml", "Le sable"

Prononciation normale du /l/ : :
"Albahr", "La mer"

3. **Suppression des lettres dupliquées sauf quelques lettres emphatiques :**
   Description :
   La deuxième règle dans l'ordre de la chaîne de traitement est la suppression des lettres dupliquées. Ces lettres sont produites par des internautes pour exprimer leurs émotions ou faire l'accentuation sur une chose importante. Si le mot contient plus de deux fois la même lettre successivement, on supprime les lettres dupliquées et on laisse que deux. Cela parce qu'en arabe, il y a des lettres emphatiques qui se prononcent avec alourdissement. Ces lettres sont au nombre de 7 [2] et en latin elles nécessitent de doubler la lettre pour la transcrire correctement. par exemple le lettre "ط" est représentée par "tt". Alors que, si le mot (token) est une suite de la même lettre comme (hhhhh) pour des rires ou (mmmmm) on le laisse tel quel, ils serviront plus tard dans la détection de sentiment.

   Exemple :
   "mliiiih !", "cooool !"

4. **Translittération des préfixes spéciaux**
   Description :
   La langue arabe étant une langue agglutinante, les articles et les connecteurs collent aux mots, comme le préfixe "بال" qui veut dire « avec un(e) » qui se compose de deux connecteurs : la conjection "ب" et l'article défini "ال". Il peut y avoir plusieurs écritures en latin pour ce préfixe "bal" ou "bel" ou "bl". Pour convertir les préfixes, on a créé trois dictionnaires : préfixe longueur 1, 2 et 3. `dico_first_one` contient aussi les cas possibles pour le préfixe " ا ", "a" : (a,o,i,u,e,é,è,2). En arabe, au début d'un mot ces lettres ne sont pas des voyelles, elle représentent une consonne. Mais au milieu d'un mot elles peuvent représenter des voyelles brèves ou des consonnes.

   Exemple :
   "belha9", "en vérité"

5. **Translittération des suffixes spéciaux**
   Description :
   De la même façon que les préfixes nous avons traité les suffixes. Les dictionnaires

1. https://en.wikipedia.org/wiki/Sun_and_moon_letters
2. Pour plus de détail : http://fi-ri7abi-l-quran.over-blog.com/article-23695515.html

(`dico_last_one, dico_last_two, dico_last_three`) contiennent la translittération des terminaisons des cas spéciaux où les voyelles (a, o, u, i) sont transcrites en voyelles longues (و، ا، ي), par exemple la conjugaison de la troisième personne du pluriel au présent qui se termine en " ون " ou la forme plurielle des noms de nombre de deux personnes "ان", "ين". La translittération de la lettre "a" à la fin des mots arabes écrits en latin est assez complexe, elle peut représenter deux lettres arabes différentes : la voyelle longue "ا" comme à la fin du pronom personnel "انا", « je » et la consonne "ة" pour marquer la fin des noms et adjectifs féminins singuliers comme "جامعة", « université ». Il n'y pas de règle morphologique pour gérer l'exception. Mais de façon générale, les mots qui sont courts prennent "ا" à leur fin comme ("لا", "كلما", "ما"...) au lieu de "ة" dans ("مدرسة", "حديقة", "اقتصادية"...). Alors, dans le système de translittération : si le mot finit par un "a" et sa longueur est inférieure à cinq lettres, il sera transcrit en "ا", sinon par "ة"

6. **Translittération des lettres en triplet**
   Description :
   Cette règle gère les semi-voyelles arabes "وا ي", lorsqu'elles sont suivies de voyelles longues, nécessitent trois lettres latines pour les transcrire comme les triplets "wai" ("وي"), "yaw" ("ياو"),"oua" ("وا"). Elle gère aussi le cas ou la lettre "a" est utilisé pour désigner la lettre "ع" comme dans "aouda", "عودة", "rentrée".

   exemple :
   "wain", "où"
   "ouahed", "un seul"

7. **Translittération des lettres en double**
   Description :
   Cette règle convertit les lettres emphatiques arabes qui se transcrivent en deux lettres latines. Elle convertit également le Hamza en respectant ses règles d'écriture en début, milieu et fin du mot et aussi les voyelles brèves qui le précèdent.

   exemple :

   a) "9issa", "قصة", "histoire"

   b) "Mala2ika", "ملائكة", "ange"

8. **Translittération des lettres uniques**
   Description :
   C'est la dernière règle en ordre de priorité. Elle consiste à convertir le restant des lettres si une ou plusieurs règles précédentes sont appliquées sur un mot ou le convertir lettre par lettre si aucune règle précédente n'est appliquée.

6.3 Évaluation

Le système de la translittération a été évalué en appliquant la distance de Levenshtein [3] entre le mot correct (référence) et le mot translittéré automatiquement (résultat). Pour avancer rapidement dans la création du corpus des mots référents, les mots sont générés automatiquement avec le système de translittération déjà existant, puis les erreurs sont corrigées

3. https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

manuellement. Dans l'évaluation on retourne la valeur de la distance Levenshtein entre les deux mots. Si la distance est nulle ; il n y a pas d'erreur et les deux mots (référence, résultat) sont identiques mais si la distance est positive ; cela veut dire il y a au moins une différence d'une lettre entre le mot référént et résultat. La table 6.1 indique la répartition des erreurs selon la distance de Levenstein.

Nombre de mots	20316
Mots corrects	72.32 %
Mots incorrects	27.68 %
Mots en erreur >= 2	10.19 %
Mots en erreur >= 3	3.54 %
Mots en erreur >= 4	1.25 %

TABLE 6.1 – Résultats de l'évaluation de la translittération

6.4 Discussions

Le système de translittération est évalué au fur et à mesure de la création des règles et les règles sont déduites à partir des erreurs du système après chaque évaluation.Nous analysons les erreurs en détail et modifions ou ajoutons des règles en conséquence dans le système afin de traiter ces erreurs. Les résultats finaux sont donc dépendants des erreurs trouvées dans le corpus.

Cette évaluation est postérieure à l'établissement de toutes les règles décrites précédemment 6.2. On constate que 72.32% des mots sont correctement translittérés, ce qui représente 14692 mots. Il reste 27.68% des mots en erreur (5624 mots). Ce taux d'erreur reste assez important, mais on analysant bien les résultats nous voyons que plus de la moitié de ces erreurs sont à une distance de 1 entre mot référent et résultat, une seule lettre est incorrecte. Ceci est pour partie lié au fait que l'arabizi n'est pas une langue standardisée : les utilisateurs du web écrivent différemment certains mots, en particulier pour les voyelles. par exemple ces deux écritures "achehal" et "echehel" qui veulent dire le même mot "اشحال" , « combien ». Par ailleurs, on remarque bien que seules 3.54% des erreurs concernent plus de trois lettres. Les erreurs sont dues aussi au cas où la règle ne prend pas correctement en charge les exceptions et translittère des mots qui satisfont certains critères mais n'auraient pas dû être translittéré selon la règle. Par exemple le cas du "a" en fin des mots qui peut avoir deux interprétations. La règle dit que si le mot contient moins de cinq lettres le "a" devient "ا" ; par contre il y a des mots qui contiennent moins de cinq lettres mais qu'il faudrait transcrire en "ه" et pas en "ا" comme "7ba", "sena", "haja".

Afin d'améliorer ces résultats, les mots que nous ne pouvons pas gérer avec des règles, nous avons décidé de créer des dictionnaires de mots avec leur translittération correspondante dans des fichiers textes externes. Ces dictionnaires contiennent aussi les mots que les utilisateurs écrivent mal couramment sur le web et leur translittération correcte. Les mots sont répartis par longueurs *length 3, 4, 5, 6*

Dans le programme, ces dictionnaires sont devenus la règle numéro deux en ordre de priorité. En lançant le programme, le système va d'abord créer des dictionnaires à partir des fichiers textes et les stocker en mémoire, ensuite vérifier si le mot est une clé du dictionnaire si oui, retourner sa valeur qui est la translittération du mot, sinon il continue à appliquer les règles de translittération suivantes sur le mot en respectant l'ordre de priorité.

length_3 :	
3ala	على
3adi	عادي
3lik	عليك
3lih	عليه
7ata	حتى
7aja	حاجة
7sal	حصل
9rit	قريت
9fas	قفص

TABLE 6.2 – Extrait du dictionnaire des mots en longueur de 3

Deuxième évaluation

Les résultats de l'évaluation après la création des dictionnaires sont ainsi :

Nombre de mots	20316
Mots corrects	75.23 %
Mots incorrects	24.77 %
Mots en erreur >= 2	7.92 %
Mots en erreur >= 3	2.02 %
Mots en erreur >= 4	0.66 %

TABLE 6.3 – Résultats d'amélioration avec les dictionnaires

On remarque une très nette amélioration dans les résultats surtout pour les erreurs dont la distance est supérieure à quatre lettres qui est à 0.66% (133 mots) dans l'ensemble des données et les erreurs sont majoritaires en distance de une lettre, cela est dû à des cas ou une seule lettre latine est utilisée pour exprimer deux ou plusieurs lettres arabes. La raison de ce choix est la différence de nombre de sons et de lettres alphabétiques entres les deux langues. Par exemple la lettre "h" est utilisée pour les deux lettres "ح", "ه" et "q" pour "ق", "ك". Le mot est uniquement distinguable en contexte. Alors, si on se permet de tolérer les erreurs de 1, on a un résultat bien meilleur, une F-mesure de 0.89.

CHAPITRE 7

LEMMATISATION

Sommaire

7.1 Méthodes **41**
Description du système existant 41
7.2 Résultats **42**
Évaluation avant l'amélioration 42
Évaluation après l'amélioration 42
7.3 Discussions **43**

7.1 Méthodes

Le mot arabe est une suite de morphèmes agglutinés les uns aux autres. Ces sont des formes assemblées (conjonction + article + radical + affixes) pour former des unités grammaticales complètes. Pour les lemmatiser, idéalement, elles nécessitent une étape de disjonction des formes au préalable, c'est-à-dire ramenée à une succession de lemmes distincts, mais concrètement, c'est la forme graphique entière qui est lemmatisée car ces flexions ont subi des modifications lors de l'agglutination des fragments, ces modifications peuvent être importantes ou faibles. *"La structure du mot arabe est donc décomposable en cinq éléments : proclitique, préfixe, base, suffixe et enclitique. La base est une combinaison de lettres radicales (le plus souvent trois) et d'un schème (redoublement, augment syllabique, voyelles longues et brèves, etc.)"*.[Tuerlinckx, 2010]

Pour cette tâche, nous avons contribué à l'amélioration du système de racinisation en corrigeant et créant quelques règles.

Description du système existant

1. Normalisation des diacritiques (les voyelles brèves " مُعَلِم" devient " معلم") et Hamza du alif (أ إ آ deviennent ا)
2. Exclure les mots vides.
3. Retourner le mot si la taille du mot est inférieure à trois lettres.
4. Racinisation des verbes faibles (verbes assimilés, verbes concaves, verbes défectueux). [1]
5. Enlever les préfixes et les suffixes.
6. Si la longueur du mot est supérieure à trois, alors trouver son schème, pour pouvoir enlever les lettres excédentaires et extraire la racine du mot.

1. Verbe assimilé : celui dont la première radicale est un و ou un ي, ex. : وصل، يَتَم
Verbe concave : celui dont la deuxième radicale est un و ou un ي, ex. : قام، باع.
Verbes défectueux : celui dont la dernière radicale est un و ou un ي , ex. : سرو ، رضي
Pour plus d'infos allez sur : ce lien

Les règles ajoutées

La racine arabe est composée exclusivement de consonnes. Elle est le plus souvent trilitère, c'est-à-dire composée de trois lettres. Le schème de la racine trilitère est "فعل" et toutes les lettres excédentaires de ce schème doivent être enlevées pour extraire la racine. Par exemple tous les mots qui ont le schème "فاعل، افعال، فواعل" deviennent "فعل". Dans l'algorithme, chaque règle traite un seul schème qui est le moule racinisant plusieurs mots de la langue arabe et les noms "صداع, سعال, زكام" par exemple possèdent le même schème "فعال" car ils se prononcent de la même manière du fait que leurs lettres sont vocalisées identiquement selon l'ordre. Pour obtenir la racine, la règle définit d'abord le schème en regardant les les lettres ajoutées, dans le cas précèdent : si la taille du mots est quatre et la troisième lettre est un "ا", on enlève le "ا" et on garde les autres lettres et ça donne les racines : "صدع, سعل, زكم".
C'est le même mécanisme pour toutes les autres règles ajoutées qui gèrent d'autres schèmes.

1. **Mots longueur 4 :**
   a) **Schème "افعل"**
2. **Mots longueur 5 :**
   a) **Schème "فعالة"**
   b) **Schème "فعلاء"**
   c) **Schème "متفعل"**
3. **Mots longueur 6 :**
   a) **Schème "مفاعيل"**
   b) **Schème "افعنلل"**
   c) **Schème "افعلال"**

7.2 Résultats

Évaluation avant l'amélioration

Nombre de mots	31352
Mots corrects	80.15 %
Mots incorrects	19.85 %
Mots en erreur >= 2	11 .21 %
Mots en erreur >= 3	5.49 %
Mots en erreur >= 4	3.15 %

TABLE 7.1 – Évaluation avant toutes modification dans le système de racinisation

Évaluation après l'amélioration

Nombre de mots	31352
Mots corrects	93.01 %
Mots incorrects	6.99 %
Mots en erreur >= 2	4.33 %
Mots en erreur >= 3	2.11 %
Mots en erreur >= 4	0.55 %

TABLE 7.2 – Évaluation après les modifications dans le système de racinisation

Rappel	0.93
Précision	0.93
F-mesure	0.93

TABLE 7.3 – Évaluation du système de racinisation

7.3 Discussions

Pour connaître le taux d'amélioration, une pré-évaluation a été faite avant les modifications et une autre après. Pour l'évaluation, une distance de Levenshtein est calculée comme précédemment dans la translittération entre le lemme référent et le lemme calculé automatiquement. On observe que les résultats sont augmentés de 13%, environ 5000 mots sont corrigés et nous sommes à 93.01% d'exactitude. Les erreurs sont estimées à 6.99%. Les limites de ce système sont produites par quelques règles trop gourmandes qui retirent les lettres agglutinées des mots dont ces lettres sont originales dans ces mots, surtout les articles et les prépositions d'une taille de une lettre comme (ب، ف، و، ل). Ainsi, tous les mots en longueur supérieure à trois et commencent par l'une de ces lettres vont la perdre et le mot reste sans valeur sémantique ou changent de sens comme les verbes "برهن، فلسف، لاعب" car lapremière lettre est originale dans le mot. Contrairement au mots "فذهب، ليجمع" où la première lettre "ف ل" représente une préposition. Comme solution à ce problème, on a proposé de créer des ressources externes pour conserver la racine des mots en exception et éviter leur déformation.

CHAPITRE 8

ANALYSE DE SENTIMENTS

Sommaire

8.1 Méthodes **45**
Ressources : 45
Approche lexicale : 46
Approche par apprentissage : 46
8.2 Résultats **47**
8.3 Discussions **47**
Résultat du Data Sampling 48

8.1 Méthodes

Ressources :

1. **Lexique orienté :** un fichier CSV comme base de donnée a été crée dans le format (ArabicWord,POS,Sentiment,English Word). Il est constitué d'un lexique de plus de 7000 termes répartis en plus de 4 catégories. Des valeurs sémantiques des mots sont proposées :
   - **Positif :** ensemble de mots positifs 2000 mots de poids +1.0
   - **Négatif :** ensemble de mots négatifs 4000 mots de poids -1.0
   - **Atténuateur :s** Adverbes réduisant l'intensité comme peu, seulement, légèrement... de poids +0.5
   - **Exhausteurs :** Adverbes augmentant l'intensité comme très, beaucoup, énormément... de poids +2.0

ArabicWord,POS,Sentiment,English Word
قليل,*,0.5,little
فقط,*,0.5 only
بشدّة,*,2,intensely
قصوى,*,2,utmost
فظاعة„-1,abomination
عدوان„-1,aggression
فوائد„+1,benefits
مريح„+1,comfy

TABLE 8.1 – Création du lexique sémantique

2. **Correction d'orientation :** Une autre base contenant des adjectifs et adverbes d'intensité mais contenant des erreurs dans leur polarité. Les érreurs ont été corrigées et les mots représentant une racine ont été signalés avec "*". Par exemple l'adjectif "شنيع" était en catégorie positive alors qu'il est négatif.
3. **Répartitions de données :** Pour rappel le corpus d'analyse de sentiment contient 1230 documents, 8423 phrases, 182783 mots. Il est divisé en 80% pour l'apprentissage et 20% pour le test. La distribution des sentiments sur la totalité des documents est la suivante, négatif : 219 documents, neutre : 7920 documents, positif : 284 document.

Approche lexicale :

Les approches lexicales dans l'analyse de sentiments sont basées sur l'utilisation d'un lexique de mots subjectifs qui ont reçu des traits spécifiques marquant le sentiment positif ou négatif. Ces mots représentent une référence universelle de sentiment dans toutes les langues et avec cette référence la tonalité peut être détectée. Ce lexique de mots peut prendre la forme de dictionnaires, une polarité étant associée à chacun des mots. Quelque soit son contexte dans lequel il apparaît, le mot aura donc constamment la même polarité. On attribue ensuite au document un score d'opinion selon la présence des mots de ces dictionnaires dans le texte.

Ces dictionnaires sont utilisés pour classifier et catégoriser des textes dont on sait qu'ils parlent d'un concept qui nous concerne, par exemple pour la e-réputation d'une voiture sur le marché, une crème de visage. La méthode consiste principalement à détecter les mots, souvent des adjectifs et adverbes qualifiants qui sont en co-occurrence avec l'unité d'appellation de cette marque, généralement dans la même phrase. Ensuite, le dictionnaire donne la tonalité attribuée à l'entité concernée par ses termes qualifiants. Il y a beaucoup de problèmes syntaxiques qui déstabilisent la relation entre une entité et ses termes qualifiants, ce qui complique la décision automatique de la polarité du sentiment détecté. C'est pourquoi d'autres traitements automatiques doivent être réalisés pour obtenir des résultats pertinents, par exemple en utilisant l'apprentissage automatique.

Limites de l'analyse de sentiment à partir des lexiques :

[Boullier and Lohard, 2012] mentionnent dans leur livre que :

1. *"Les dictionnaires affectent une tonalité positive ou négative à un mot, sans tenir compte du contexte, c'est-à-dire du texte environnant, comme des paramètres de la communication située"*.
2. *"Les dictionnaires ont tendance à éliminer les mots à valence ambiguë a priori ; le traitement des expressions ambiguës reste à faire et demande de faire appel à d'autres principes et à d'autres techniques"*.
3. *"Ces dictionnaires ne permettent pas de traiter des figures de rhétorique qui peuvent changer entièrement la valence des expressions (le sarcasme, l'ironie : « encore une belle réussite de notre super président ! »)"*.

Approche par apprentissage :

SGDClassifier : Cette phase a été réalisé en utilisant l'algorithme d'apprentissage authomatique "sklearn" [1] avec son classifieur linaire "SGDClassifier" adapté aux besoins de l'équipe data science chez Synthesio pour l'analyse de sentiment. À partir des données que j'avais annotées pour l'analyse des sentiments, Synthesio a créé avec *SGDClassifier* un modèle de donnée que j'améliore après chaque correction d'erreurs.

1. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

8.2 Résultats

Après chaque exécution du modèle sur le données, un fichier d'erreurs est généré puis corrigé manuellement pour améliorer les résultats avec le nouvel apprentissage. Le cycle de correction est défini en trois phases : **1- nouveau modèle, 2- nouvelle liste de mots en erreurs, 3- correction**. A chaque fois on donne un nouveau fichier le système apprend davantage et les résultats de détection de sentiment s'améliorent. L'opération a été répétée une dizaine de fois et les 10 fichiers sont corrigés.

1. **Table de confusion après la 5ème optimisation :**
   (ligne = référence ; colonne = hypothèse)

	-	0	+	Total
-	$< 36 >$	162	21	219
0	394	$< 7147 >$	379	7920
+	13	200	$< 71 >$	284
Total	443	7509	471	

TABLE 8.2 – Résultats de détection de sentiment après la 5ème correction

Rappel	0.88
Précision	0.86
F-mesure	0.87

2. **Table de confusion après la 10ème optimisation :**
   (ligne = référence ; colonne = hypothèse)

	-	0	+	Total
-	$< 46 >$	144	29	219
0	337	$< 7389 >$	194	7920
+	28	169	$< 87 >$	284
Total	411	7702	310	

TABLE 8.3 – Résultats de détection de sentiment après la 10ème correction

Rappel	0.90
Précision	0.89
F-mesure	0.89

8.3 Discussions

On considérant les résultats, on remarque après la cinquième correction que les vrais positifs (entre chevrons) sont estimés à 7254 sur une totalité de 8423 et après la dixième sont estimés à 7522. Donc, le silence concerne 9% des données pertinentes. Beaucoup de documents neutres sont classés en négatif, 394 documents et 379 en positifs dans la cinquième correction mais on aperçoit une amélioration dans la dixième surtout pour les positifs, avec une diminution importante à 194 et une autre moins en négatifs à 337. Ces erreurs sont généralement des substantifs qui sont liés au nom propre d'une marque précise. Le système lui a attribué un score négatifs à cause de ses co-occurrents négatifs. Par exemple : « cette maxima est très chère ! » (très chère) est négatif donc il apprend que partout où il y a la voiture maxima c'est négatif. Aussi, la distribution des sentiments sur le nombre de documents

pause un problème. Elle n'est pas homogène, plus de 60% des données sont neutres. Dans ce cas il y a beaucoup plus d'exemple sur le neutre. Ainsi, le système a appris davantage pour détecter le neutre et il n'a pas assez d'exemples sur le négatif et le positif.

Pour cela on a décidé de faire un sampling pour les données et de faire une nouvelle répartition de façon qu'elle soit plus au moins équilibrée entre (positif, négatif, neutre). C'est à dire augmenter le nombre de documents positifs, négatifs et diminuer celui des neutres. Une autre solution supplémentaire était de créer des nouveaux fichiers des mots mal classifiés dont les mots portent une polarité erronée. Il s'agit de corriger leur polarité en les classant dans 6 catégories différentes :

1. Positive considered as Neutral
2. Positive considered as Negative
3. Neutral considered as positive
4. Neutral considered as Negative
5. Negative considered as Neutral
6. Negative considered as positive

Dans chaque catégorie de ces 6 catégories, il y a deux autres catégorie, lemma pour les mots qui sont des racines et word pour le cas contraire.

Positive considered as Neutral :
lemma :
خير
ملك
عجب
word :
نوعية
تبارك
التقييم

TABLE 8.4 – Correction de polarité des mots

Résultat du Data Sampling

(ligne = référence ; colonne = hypothèse)

	-	0	+	Total
-	$< 321 >$	61	56	438
0	64	$< 2229 >$	51	2344
+	13	24	$< 531 >$	568
Total	302	2639	409	

TABLE 8.5 – Résultats de détection de sentiment après le sampling

Rappel	0.92
Précision	0.91
F-mesure	0.91

Les tableaux des résultats du sampling montrent une amélioration positive mais pas très significative. Un gain de 170 vrais positifs sur une totalité de 8423, ce qui veut dire une augmentation de 2%. Pour cela, Nous pouvons supposer que ces faibles gains sont liés à la taille insuffisante des fichiers de correction de polarité qui attendent un enrichissement futur.

CONCLUSION GÉNÉRALE

Conclusion

Dans ce mémoire, nous avons traité les problématiques liées à la translittération et à la racinisation de la langue arabe pour l'analyse des sentiments dans les médias sociaux. Ces trois thèmes différents translittération, racinisation et analyse des sentiments sont des sujets interconnectés entre eux implémentés dans une chaîne de traitement qui a pour objectif final la détection de sentiment des messages issus du web social pour la e-réputation au service des marques. Le but de la translittération est de traiter la langue arabe écrite en latin par des internautes de la communauté arabe qui préfèrent utiliser ce format à la place de l'arabe moderne standard. Ceci facilite par la suite le traitement automatique de ce type de langue et l'application de systèmes de racinisation et de détection de sentiment.

Pour l'identification et la translittération de l'arabizi, on a entraîné et testé des données issues du web et atteint un résultat de 96% pour l'identification de l'arabizi en machine learning (la bibliothèque keras) et 75% en translittération lorsque l'on omet les erreurs ne portant que sur une lettre par mot (distance de levenshtein). La translittération a été réalisée en construisant des règles de transformation appliquées par ordre de priorité. Une fois le mot translitteré, on aura besoin de trouver sa racine pour y appliquer d'autres règles de détection de sentiment. Le système de racinisation est aussi un système à base de règles qui permet de séparer les racines de leur préfixes et suffixes. On a eu un résultat de 93% de précision. L'analyse de sentiment effectuée à l'aide de deux approches : lexicale et apprentissage automatique (en utilisant le classifieur SGD) avec un résultat de 90% de rappel et 89% de précision.

Perspectives

Dans ce travail, on a traité de l'arabe dialectal avec toutes ses variétés malgré les différences phonétiques et morphologiques entre elles qui sont des limites importantes pour notre système. Pour un futur travail, on souhaite traiter chaque dialecte singulièrement par le développement d'un système d'identification des dialectes arabes et constituer chacun ses ressources lexicales uniques. Nous pouvons aussi envisager, comme autre alternative, convertir du dialectal vers MSA avant tout traitement automatique puisqu'il n'existe aucune nome d'orthographe pour les dialectes. Comme perspective à la translittération, je propose d'intégrer la vocalisation (les voyelles brèves) dans la transformation des mots pour distinguer leur sens et catégorie grammaticale.

BIBLIOGRAPHIE

[Al-Badrashiny et al., 2014] Al-Badrashiny, M., Eskander, R., Habash, N., , and Rambow, O. (2014). Automatic transliteration of romanized dialectal arabic. in proceedings of the conference on computational natural language learning (conll). *. – Cité pages 18, 25 et 26.

[Al-Onaizan and Knight, 2002] Al-Onaizan, Y. and Knight, K. (2002). Machine transliteration of names in arabic text. *, pages 1–13. – Cité page 24.

[Arababi et al., 1994] Arababi, M., Fischthal, S. M., Cheng, V. C., and Bar, E. (1994). Algorithms for arabic name transliteration. ibm journal of research and development. *. – Cité page 23.

[Bies et al., 2014] Bies, A., Song, Z., Maamouri, M., Grimes, S., Lee, H., Wright, J., Strassel, S., Habash, N., Eskander, R., and Rambow, O. (2014). Transliteration of arabizi into arabic orthography: Developing a parallel annotated arabizi-arabic script sms/chat corpus. *. – Cité pages 25 et 28.

[Boullier and Lohard, 2012] Boullier, D. and Lohard, A. (2012). *OPINION MINING ET SENTIMENT ANALYSIS Chapitre 5. Détecter les tonalités : opinion mining et sentiment analysis*. OpenEdition Press. – Cité page 46.

[Buckwalter, 1990] Buckwalter, T. (1990). Arabic transliteration. *. – Cité page 23.

[Chalabi and Gerges, 2012] Chalabi, A. and Gerges, H. (2012). Romanized arabic transliteration. *. – Cité page 24.

[Cherry and Suzuki, 2009] Cherry, C. and Suzuki, H. (2009). Discriminative substring decoding for transliteration. *journal*. – Cité page 24.

[Darwish, 2013] Darwish, K. (2013). Arabizi detection and conversion to arabic. *. – Cité page 24.

[Federico et al., 2008] Federico, M., Bertoldi, N., and Cettolo, M. (2008). Irstlm: an open source toolkit for handling large scale language models. *. – Cité page 25.

[Habash, 2010] Habash, N. (2010). Orthographic and morphological processing for english-arabic statistical machine translation. *. – Cité page 24.

[Habash et al., 2012] Habash, N., Diab, M., and Rambow, O. (2012). Conventional orthography for dialectal arabic. *. – Cité page 25.

[Kahki et al., 2011] Kahki, A. E., Darwish, K., Din, A. S. E., and El-Wahab, M. A. (2011). Transliteration mining using large training and test sets. *. – Cité page 25.

[Och, 2003] Och, F. J. (2003). Minimum error rate training in statistical machine translation. *. – Cité page 24.

[Sherif and Kondrak, 2007] Sherif, T. and Kondrak, G. (2007). Substring-based transliteration. *. – Cité page 24.

[Stalls and Knight, 1998] Stalls, B. G. and Knight, K. (1998). Translating names and technical terms in arabic text. in proceedings of the coling/acl workshop on computational approches to semitic languages. *. – Cité page 23.

[Synthesio, 2014a] Synthesio (2014a). Guide d'utilisation des dashboards. *. – Cité page 16.

[Synthesio, 2014b] Synthesio (2014b). *Guide d'utilisation Le SETUP*. Paris, France. – Cité page 16.

[Tobaili, 2016] Tobaili, T. (2016). Arabizi identification in twitter data. *. – Cité pages 18 et 20.

[Tuerlinckx, 2010] Tuerlinckx, L. (2010). La lemmatisation de l'arabe non classique. *. – Cité page 41.

ANNEXES

A.1 Exemple de code d'une règle de translittération

Suppression des lettres dupliquées sauf quelques lettres emphatiques

```
    # Rule 3 remove duplicated letters
    # List of letters that can be paired
    paired_letters = ['a', 'o', 'e', 'i', 's', 't']
    first_let = token[0]
    cpt = 0
    # Check if all letters are the same
    for let in token:
        if let != first_let:
            # There is difference --> stop loop
            break
        cpt += 1
    # All letters are same? --> Don't remove duplicate (examples: hhhhhh,
       mmmm)
    if cpt == len(token):
        return token, ''
    # Remove duplicates
    result = ''
    prefix = ''
    i = 0

    # Ignore duplicated letters
    while i < len(token):
        result += token[i]
        letter = token[i]
        i += 1
        # Letter that can be paired --> if duplicated then duplicate it
           once
        if (i < len(token)) and (letter in paired_letters) and (letter ==
           token[i]):
            result += letter
            i += 1
        # Same letter --> ignore it
        while i < len(token) and token[i] == letter:
            i += 1

    if prefix == '':
        return (result, prefix)
    else:
```

```
        # ignore first letter of the remained token which is already
            converted in the prefix
        return (result[1:], prefix)
```

A.2 Code de quelques règles de racinisation ajoutées

Schème "افعل"

```
elif self.stm[0] == u'ا':  #new pattern افعل
    self.pttrn = u'افعل'
    self.stm = self.stm[1:]
    return self.stm
```

Schème "فعلاء"

```
elif ((self.stm.endswith(u'اء'))):        #new pattern فعلاء
    self.pttrn = u'فعلاء'
    self.stm = self.stm[:-2]
    return self.stm
```

Schème "مفاعيل"

```
elif (self.stm[0]== u'م' and self.stm[2]== u'ا' and self.stm[4]== u'ي'):
   #new pattern مفاعيل
    self.pttrn = u'مفاعيل'
    self.stm = self.stm[1]+self.stm[3]+self.stm[5]
    return self.stm
```

A.3 Script d'évaluation avec la distance de Levenshtein

```
# -*- coding: utf-8 -*-
'''
Created on 24 mai 2016
@author: Lilia
'''
import os
import codecs
import unicodedata as ud
from synthesio.algorithm.language.transliteration.arabicTransliteration
   import latinToArabic
from synthesio.data.dictionary.arabicAutoLemmaPos import ArabicAutoLemmaPos

latin_letters= {}

def is_latin(uchr):
    try: return latin_letters[uchr]
    except KeyError:
        return latin_letters.setdefault(uchr, 'LATIN' in ud.name(uchr))

def only_roman_chars(unistr):
    return all(is_latin(uchr)
           for uchr in unistr
```

```
            if uchr.isalpha())


def levenshteinDistance(s1, s2):
    m=len(s1)+1
    n=len(s2)+1

    tbl = {}
    for i in range(m): tbl[i,0]=i
    for j in range(n): tbl[0,j]=j
    for i in range(1, m):
        for j in range(1, n):
            cost = 0 if s1[i-1] == s2[j-1] else 1
            tbl[i,j] = min(tbl[i, j-1]+1, tbl[i-1, j]+1, tbl[i-1,
                j-1]+cost)

    return tbl[i,j]

distances = []
len_words = 0
nmbr_words = 0
correct_words = 0
non_correct_words = 0
for root, dirs, files in os.walk("/Users/nouveau/dev/evaluation"):
    for name in files:
        if name[0]!='.':
            fullpath = (os.path.join(root, name))
            input_file = codecs.open(fullpath, 'r', encoding='utf-8')
            for line in input_file.readlines():
                if len(line) > 0 and
                    only_roman_chars(unicode(line[0]))==True:
                    parts = line.split('\t')
                    if len(parts) > 1 and
                        only_roman_chars(unicode(parts[1]))==False :
                        nmbr_words += 1
                        trans = latinToArabic(parts[0])
                        if parts[1]==trans:
                            correct_words += 1
                        else:
                            non_correct_words += 1
                            #incorect_words.write('\''+parts[0]+'\', ')
                            #dico[parts[0]]+=1

                        leven_distance = levenshteinDistance(trans,
                            parts[1])
                        distances.append(leven_distance)
                        #dico[parts[0]]+=1

print 'Le nombre de mots: ',nmbr_words
print 'Les mots corrects: ',correct_words, ' donc: ',
    round(float((correct_words*100))/nmbr_words, 2), '%'
print 'Les mots incorrects: ',non_correct_words, ' donc: ',
    round(float((non_correct_words*100))/nmbr_words, 2), '%'

error1 = len([val for val in distances if val >=1])
```

```
print'Les mots en erreur 1 et plus: ', error1, ' donc: ',
    round(float((error1*100))/nmbr_words, 2), '%'
error2 = len([val for val in distances if val >=2])
print'Les mots en erreur 2 et plus: ', error2, ' donc: ',
    round(float((error2*100))/nmbr_words, 2), '%'
error3 = len([val for val in distances if val >=3])
print'Les mots en erreur 3 et plus: ', error3, ' donc: ',
    round(float((error3*100))/nmbr_words, 2), '%'
error4 = len([val for val in distances if val >=4])
print'Les mots en erreur 4 et plus: ', error4, ' donc: ',
    round(float((error4*100))/nmbr_words, 2), '%'
```