Ministère de l'enseignement supérieur et de la recherche scientifique
UNIVERSITE SAAD DAHLEB DE BLIDA 1
Faculté des Sciences
Département d'Informatique

MÉMOIRE DE MASTER INFORMATIQUE TAL

Généralisation d'approches basées Word Embedding pour un système d'évaluation des réponses courtes adapté à la langue anglaise.

Réalisé par :

BENTEKKOUKA Abderrahmane
CHEIK Abdellah

Proposé et encadré par :

Mme OUAHRANI Leila

Composition de jury :

M. FERFERA SOFIANE Président
Mme. ZAHRA Examinateur

Soutenu le :
9 Janvier 2020

ملخص

يتمثل العمل المقدم هنا في تكرار طريقة حساب التشابه المتماثلة المقترحة في دراسة سابقة تم إجراؤها للغة العربية والتي تألفت من قياس تأثير "Word Embeddings" على أنظمة التقييم التلقائي للإجابة القصيرة ، لاختبار تأثير تضمين كلمة في اللغة الإنجليزية ، من الواضح أنها اكتسبت جميع الموارد اللازمة مثل stemmers، corpus، WE .. التعامل مع اللغة الإنجليزية ، وبالتالي لمعرفة إمكانية نع أو رؤية جدوى التعميم لهذا النظام

الكلمات المفتاحية : تقييم تلقائي لإجابات قصيرة، مقاييس تشابه، معالجة الآية لغة، تجذيع.

Résumé

Le travail présenté ici consiste à reproduire la méthode du calcul de similarité proposée dans une étude précédente réalisée pour la langue arabe et consistant à mesurer l'effet de "Word Embeddings" sur les systèmes d'évaluation automatique à réponse courte, afin de tester l'impact de l'incorporation de mots sur la langue anglaise, ayant évidemment acquis toutes les ressources nécessaires telles que les stemmers, corpus, WE .. traitant de la langue anglaise, et voir ainsi la possibilité de faire ou de voir la faisabilité d'une généralisation de ce système à une autre langue.

Mots clés : Evaluation automatique des réponses courtes, ASAGS, Word Embedding, Mesures de similarité, Traitement automatique de la langue, Stem, modèle d'espace vectoriel

Abstract

The work presented here is to replicate the similarity calculus method proposed in a previous study that was done for the Arabic language that consisted of measuring the effect of "Word Embeddings" on automatic short answer evaluation systems, for to test the impact of word embedding on the English language, obviously having acquired all the necessary resources like stemmers, corpus, WE .. dealing with the English language, and thus to see the possibility of making or seeing the feasibility of a generalization for this system

Keywords: Automatic short answer grading, ASAGS, Word Embedding, Similarity Measures, Natural language processing, Stem, Vector space model

REMERCIEMENTS

Nous tenons tout d'abord à remercier Allah le tout puissant, qui nous a donné la force, la capacité et surtout la patience pour accomplir ce travail.

Nous tenons à remercier tout particulièrement et à témoigner vivement toute notre reconnaissance à notre promotrice Mme Ouahrani Leila pour tous les efforts qu'elle a fournis, de s'être investi corps et âme avec rigueur scientifique pour la direction de ce mémoire. Ses qualités pédagogiques remarquables nous ont permis de profiter de ses connaissances et ont contribué à l'avancement de notre travail, même si on n'a pas été à la hauteur parfois … Merci !

Nous remercions nos très chers parents, qui ont toujours été là pour nous, « Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous nous avez donné un magnifique modèle de labeur et de persévérance. Nous sommes redevables d'une éducation dont nous sommes fiers ».

Nous tenons à exprimer nos sincères remerciements à nos amies Adel, Hamza, Oussama, Walid, Abdou, pour leur contribution, soutien et encouragement qu'avec, nous avons pu surpasser des périodes de stress

A nos compagnons de lutte universitaire, et spécialement ceux de TAL pour le courage et l'amour qu'ils nous ont caractérisés durant tout ce temps de vie estudiantine.

En fin que les amis, frères et sœurs dont les noms ne sont pas cités ne nous tiennent pas rigueur, nos pensées vont aussi vers eux.

Table des figures

Liste des tableaux

Table des matières

Table des abbreviations:

Abréviation	Description
-WE	-Word Embedding
-CP	-Coefficient de corrélation de pearson
-RMSE	-Root Mean Squared Error
-ASAGs	-Automatic Short Answer Grading -Systems
-MO	-Matrice d'ordre
-POS	-Part of Speech
-CBOW	-Continious bag of words
-TAL	-Traitement automatique de la langue

Introduction générale

1. Introduction

Aujourd'hui on constate qu'il y a de plus en plus de développement d'outils d'aide à la prise de décision par rapport à l'évaluation automatique en matière d'enseignement, Les objectifs de ces dernier allant de la réalisation et la consolidation des avantages d'un système présentant les caractéristiques primordiaux de réduction de la charge de travail des enseignants en automatisant une partie de leur tâche essentiel d'évaluation, fournir aux apprenants des informations détaillées sur leur période d'apprentissage de manière plus efficace que l'évaluation traditionnelle, à intégration la culture d'évaluation au travail quotidien des apprenants dans un environnement d'e-Learning.

Dans ce travail nous allons nous basé sur les questions ouvertes à réponses courtes dans le but de demander à l'apprenant de construire une réponse plutôt que de choisir parmi un certain nombre d'options prédéterminées, pour qu'il puisse reproduire les connaissances déjà acquises.

L'automatisation de l'évaluation de ces réponses n'est pas simple en raison de variations linguistiques (une réponse donnée pourrait être articulée de différentes façons), nature subjective de l'évaluation (multiples réponses possibles), manque de cohérence dans la notation humaine, … etc.

Le travail est orienté vers la langue arabe qui, bien que largement utilisée aujourd'hui, n'a pas encore bénéficié de recherche et de résultats matures dans le domaine de l'évaluation automatique.

2. Problématique

Le principe général est d'obtenir un score généré par un système qui a pour objectif la comparaison entre deux réponses l'une modèle (référence) formuler par l'enseignant et l'autre fournis par l'apprenant.

La motivation qui a fait que ce travail a était déclencher est la présence d'un travail monté a fin de traiter la langue arabe, où des approches statistiques d'évaluation automatique des réponses courtes utilisant une distribution multidimensionnelle (les Word Embedding) ont été développé pour faciliter le traitement de l'arabe, une langue jugé ambiguë où nos camarades ont trouvé des manques considérables du point de vue ressources linguistique (corpus arabes, lexiques et dictionnaires, outil de traitement …).

Le but est donc de répondre à la question de savoir à quel point le travail effectué est générique et indépendant de la langue traiter pour être appliqué a toute langue souffrante des mêmes manques et défis que la langue arabe, Notre choix a était porté vers la langue anglaise vue la disponibilité des ressources précédemment évoqué.

3. Objectifs

- Généraliser les approches déjà développées pour la langue arabe à l'anglais en considérant les spécificités de l'anglais et développer les outils adéquats,

- Evaluer le degré de généricité des approches développées en validant par des data sets et des métriques de performance

4. Plan de travail

- **Partie 1 :** Etat de l'art sur les outils et approches d'évaluation automatiques des réponses courtes ainsi que les Word Embedding dans le contexte de l'évaluation automatique des réponses courtes.

- **Parie 2 :** Acquisition des Word Embedding pour la distribution des mots en anglais ainsi que des corpus pour le calcul des poids des mots
- **Partie 3 :** L'etude et l'analyse des approches déjà développées pour la langue arabe,
- **Partie 4 :** Généralisation des approches sur l'anglais et développement des outils correspondants
- **Partie 5 :** Acquisition de data set et évaluation et comparaison des résultats obtenus par rapport à deux métriques : le coefficient de Pearson et l'erreur quadratique entre les notes obtenues manuellement (par l'expert humain) et les notes générés automatiquement.

5. Structure du mémoire

Le reste de notre document est structuré en 3 chapitres :

- Dans le chapitre un, nous présentons l'état de l'art du domaine pour mieux situer notre travail.

- Dans le chapitre deux, nous expliquons les méthodes des approches utilisées ainsi que la description des différentes ressources dont : les WE, les corpus, les stem … etc

- La synthèse expérimentale que nous avons réalisé est décrite dans le chapitre trois où nous présentons une évaluation des approches. Nous clôturons le chapitre par une discussion sur les résultats obtenus.

- Enfin, la conclusion générale nous permet de mettre le point sur le travail réalisé ainsi que ses perspectives futures.

Chapitre **1**

Etat de l'art

Notre travail repose sur la continuité et la généralité d'une étude similaire déjà faites sur la langue arabe, de ce fait, il se situe à l'intersection de plusieurs domaines de recherche à savoir le domaine de similarité de textes, celui des systèmes d'évaluation automatique, des Word Embeddings ainsi que le domaine du TAL. Ainsi cet état de l'art nous permet de présenter le travail qu'on souhaite adapter pour la langue anglaise, revoir les différentes approches de calcul de similarité de textes déjà développées, et introduire les Word Embeddings avec leurs différents modèles de génération.

1. Les systemes "ASAG " Automatic short answer grading-system

1.1. Definition:

La notation automatique des réponses courtes (ASAG) consiste à évaluer les réponses courtes en langage naturel à des questions objectives [1] à l'aide de méthodes de calcul. La recherche active dans ce domaine a considérablement augmenté ces derniers temps avec plus de 80 articles répondant à une définition d'ASAG. Cependant, les efforts passés ont généralement été ponctuels et non comparables jusqu'à récemment, d'où la nécessité d'une vision unifiée de l'ensemble du domaine [2]

La différence entre, les questions à choix multiples et les questions à réponse courte est facile à comprendre, mais la différence entre d'autres types de questions, telles que les réponses courtes et les dissertations, peut devenir floue. Par conséquent, nous disons qu'une question à réponse courte est une question pouvant être considérée comme répondant à au moins cinq critères spécifiques. Premièrement, la question doit exiger une réponse qui rappelle des connaissances externes au lieu d'exiger que la réponse soit reconnue à partir de la question. Deuxièmement, la question doit exiger une réponse en langage naturel. Troisièmement, la longueur de la réponse doit être comprise approximativement entre une phrase et un paragraphe. Quatrièmement, le degré d'ouverture dans les réponses ouvertes par opposition aux réponses fermées devrait être limité par une conception de question objective. Cinquièmement, les ASAGS se basent sur le contenu plutôt que sur le style. Une mauvaise qualité d'écriture (de formulation) est, jusqu'à un certain point, tolérée facilitant ainsi la production d'une idée.

1.2. Pipeline de développement de système ASAG

La notion de pipeline est bien soutenue par plusieurs domaines de la recherche en traitement du langage naturel, comme il a été démontré par Wachsmuth dans divers de ses travaux tel que l'extraction de relation et le remplissage de Template [3] ou l'extraction efficace de l'information [4]. Pour les systèmes ASAGS, la forme générale d'un pipeline de développement est constituée de 5 processus et 6 artéfacts comme le montre la figure 1 :

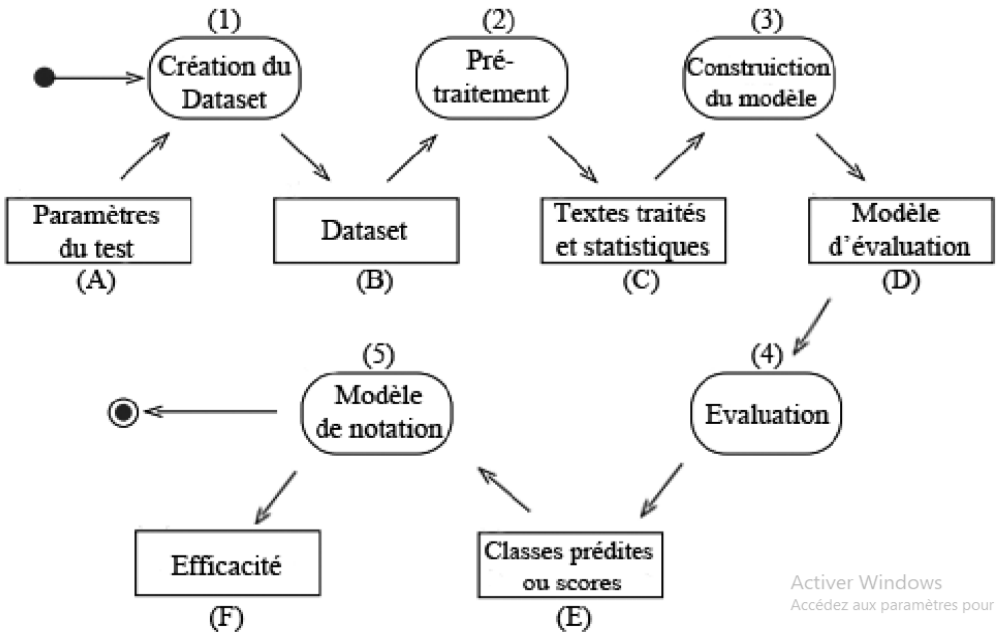

Figure 1 Pipeline de développement de système ASAG représenté par 6 artefacts (rectangles) et 5 processus (ovales) [5]

Le fonctionnement général du pipeline est relativement simple :

1) Création du dataset(Ensembles de tests) : Il est nécessaire pour la création du dataset de bien définir le contexte et les besoins du teste comme le domaine d'évaluation, la langue, et les propriétés des questions... . Une fois cette étape (A) terminée, le dataset est créé en regroupant chaque couple de « RM, RA » (réponse modèle, réponse de l'apprenant) pour avoir un dataset organisé (B).

2) Prétraitement : Ici on fait appel aux techniques de traitement automatique des langages naturels tel que la normalisation (regroupement des différentes formes que peut revêtir un mot, soit : le nom, le pluriel, le verbe à l'infinitif …) pour générer du texte comprenant des formes de mots normalisées de la façon souhaitée et générer des statistiques (C).

3) Construction du modèle : La construction du modèle est une des étapes les plus importantes du pipeline étant donné que le modèle construit(D) est l'entité chargée du calcul de similarité entre les réponses modèles et les réponses des apprenants. L'approche et les mesures de similarité à utiliser sont définies et implémentées au cours de cette étape et certaines connaissances du domaine ou ressources peuvent être requises.

4) Evaluation : Une fois le modèle construit et le dataset préparé, l'évaluation des réponses est effectuée générant un ensemble de similarités ou de classes(E).

5) Modèle de notation : Les prédictions issues de l'étape d'évaluation sont généralement des similarités et non pas des notes à proprement parlé. Pour assurer un passage vers des notes fiables, un module de notation est construit en se basant sur des algorithmes de classification. Dès lors, des comparaisons entres les notes manuelles et les notes du système sont faites afin de calculer l'efficacité de ce dernier(D).

2. Les approches de mesures de similarité

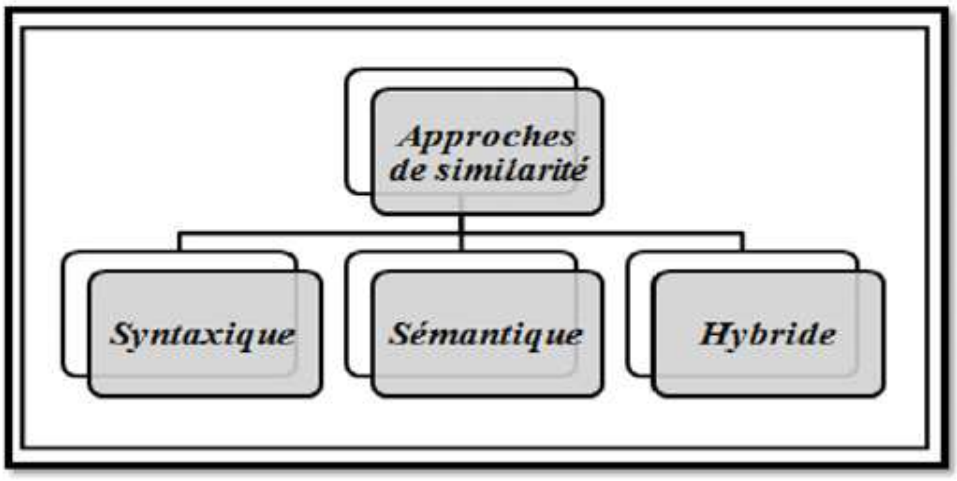

Figure 2 Les approches de similarités

La notion de similarité a été très tôt perçue comme un concept clé en intelligence artificielle ainsi qu'elle intervient dans plusieurs de ses domaines : l'apprentissage automatique, la recherche d'information, la détection de fraudes (l'empreinte digitale), la détection du plagiat, la traduction automatique des corpus, le résumé de texte…

Il existe trois catégories principales des approches de similarité (voir Figure 2) :

1. Similarité syntaxique (String-based similarity).

2. Similarité sémantique (Semantic similarity).

3. Similarité hybride (hybrid similarity)

2.1. Similarité syntaxique :

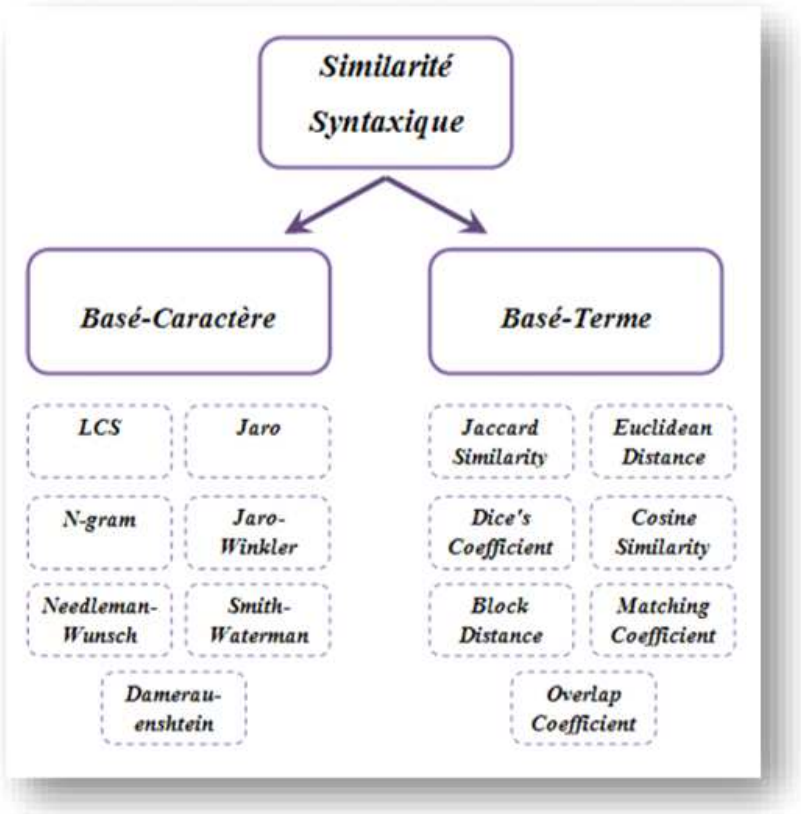

Figure 3 Mesures de similarités syntaxiques [6]

En mathématiques et en informatique, une mesure permettant de comparer des documents textuels, consiste à comparer des chaines de caractères. C'est une métrique qui mesure la similarité ou la dissimilarité entre deux chaînes de caractères. Par exemple, les chaînes de caractères "voiture" et "voiturier" peuvent être considérées comme similaires, d'autre part,

‘’voiture’’ et ‘’véhicule’’ ne le sont pas. Une telle mesure sur les chaines de caractères fournit une valeur obtenue algorithmiquement [7]. « Figure 3 » montre les mesures de similarité syntaxique. Parmi ses mesures, sept entre elles sont basées sur des caractères tandis que les autres sont des mesures de distance basée sur les termes. Parmi ces mesures nous présentons :

▪ **Indice de Jaccard** : L'indice de Jaccard (ou coefficient de Jaccard) est le rapport entre le cardinal (la taille) de l'intersection des ensembles considérés et le cardinal de l'union des ensembles [8]. Il permet d'évaluer la similarité entre les ensembles. Soit deux ensembles A et B, l'indice est $\boldsymbol{J(A,B)} = \frac{(A \cap B)}{(A \cup B)}$

▪ **Distance euclidienne :** La distance euclidienne calcule la similarité entre deux documents d1 et d2 comme la distance entre leurs représentations vectorielles ramenées à un seul point [7].

$$\boldsymbol{Simeuclidienne} = \sqrt{\sum_{i=1}^{n} (d1_i - d2_i)}$$

Où n est le nombre total de termes représentés, i.e. la taille des vecteurs.

▪ **Cosinus :** La similarité cosinus est fréquemment utilisée [9] en tant que mesure de ressemblance entre deux documents d1 et d2. Il s'agit de calculer le cosinus de l'angle entre les représentations vectorielles des documents à comparer. La similarité obtenue $simcosinus(d1, d2) \in [0; 1]$ [7].

▪ **Indice de Dice :** L'indice de Dice mesure la similarité entre deux documents d1 et d2 en se basant sur le nombre de termes communs à d1 et d2 [7].

SimDice(p1, p2) = $\frac{2N_c}{N_1+N_2}$

Où Nc est le nombre de termes communs à d1 et d2, et N1 (resp. N2) est le nombre de termes de d1 (resp. d2).

▪ **Coefficient de corrélation de Pearson :** Le coefficient de corrélation de Pearson calcule la similarité entre deux documents d1 et d2 comme le cosinus de l'angle entre leurs représentations vectorielles centrées-réduites. La similarité obtenue

$simpearson(d1; d2) \in [-1; 1]$ [7]. $\boldsymbol{simpearson}(\boldsymbol{d1}, \boldsymbol{d2}) = \boldsymbol{simcosinus}\left(\boldsymbol{d1} - \overline{\boldsymbol{d1}}, \boldsymbol{d2} - \overline{\boldsymbol{d2}}\right)$

D'où $\overline{d1}$ (resp. $\overline{d2}$) représente la moyenne de d1 (resp. d2)

▪ **Mesures de similarité basées sur le caractère**

- **LCS** [10](Longest Common SubString) est un algorithme qui considère la chaine commune la plus longue. La plus longue sous-séquence commune à deux suites, ou deux chaînes de caractère, est une séquence étant sous-suite des deux suites, et étant de taille maximum. La résolution de ce problème peut être obtenue par programmation dynamique.

- **Damerau-Levenshtein** [10] aussi connue sous le terme distance d'édition il considère la distance entre deux chaînes en comptant le nombre minimum d'opérations nécessaires pour transformer une chaîne en une autre, l'opération est définie comme une insertion, une suppression ou une substitution d'un seul caractère.

- **Bigram** [10]est une séquence de deux éléments adjacents d'une chaîne de jetons, qui sont généralement des lettres, des syllabes ou des mots. Un bigram est un n-gramme pour n = 2. La distribution de fréquence de chaque bigram dans une chaîne est couramment utilisée pour l'analyse statistique simple du texte dans de nombreuses applications, y compris en linguistique computationnelle, cryptographie, reconnaissance vocale, et ainsi de suite. dans le cas de similarité, on considère les bigrams en commun entre les termes3.

- **Trigram** [10] est une séquence de deux éléments adjacents d'une chaîne de jetons, qui sont généralement des lettres, des syllabes ou des mots. Un trigram est un n-gramme pour n = 3. La distribution de fréquence de chaque trigram dans une chaîne est couramment utilisée pour l'analyse statistique simple du texte dans de nombreuses applications, y compris en linguistique computationnelle, cryptographie, reconnaissance vocale, et ainsi de suite. dans le cas de similarité, on considère les trigrams en commun entre les termes.

- **Smith-Waterman** [11] Il effectue un alignement local pour trouver le meilleur

alignement sur le domaine conservé de deux séquences. Il est utile pour les séquences dissemblables qui sont suspectées de contenir des régions de similarité ou des motifs de séquence similaires dans leur contexte de séquence plus grand

2.2. Similarité sémantique

« La similarité sémantique est un concept selon lequel un ensemble de documents ou de termes se voient attribuer une métrique basée sur la ressemblance de leur signification ou contenu sémantique. » [6]. Ces approches sont les meilleurs pour ce qui est de capturer le sens des textes. Dans un cadre où le but est d'évaluer les compétences d'un individu dans un domaine précis (évaluation par écrit), les mesures de similarité sémantique excellent remarquablement par rapport aux mesures syntaxiques. Nous distinguons deux sous catégories (figure 4) :

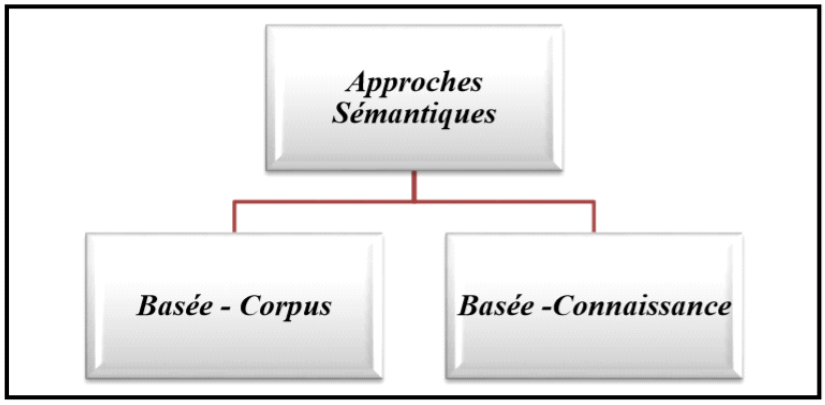

Figure 4 Sous catégories de l'approche sémantique

Similarité basée corpus

Du terme anglais « Corpus-Based Similarity », la similarité basée corpus est un ensemble de mesures de similarité sémantique qui déterminent la similitude entre les mots et donc entre les textes en fonction de l'information obtenue depuis de grands corpus. Un Corpus est une grande collection de textes écrits ou parlés utilisés pour la recherche linguistique. La figure 5 montre les mesures de similarité basées corpus les plus répondues.

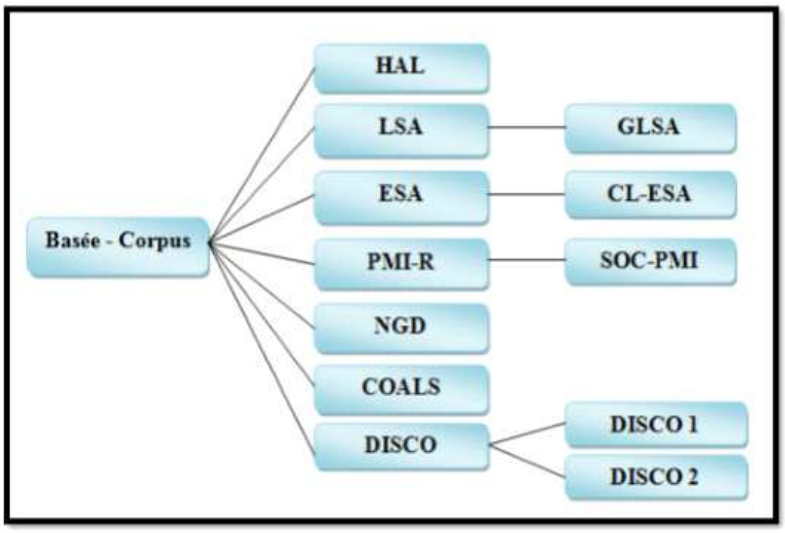

Figure 5 Les mesures de similarité basées corpus les plus répondues [12]

Similarité basée connaissance

Du terme anglais « Knowledge-Based Similarity », la similarité fondée sur la connaissance est l'une des mesures de similarité sémantique qui repose sur l'identification du degré de similitude entre les mots à l'aide d'informations dérivées de ressources externes tel que le réseau WordNet ou les dictionnaires de synonymes. La figure 6 comporte quelques mesures de similarité basée connaissance. Nous remarquons que les mesures sont classées dans deux sous branches : les mesures de similarité sémantique et les mesures de relation sémantique, ces dernières couvrent un plus large éventail de relations entre concepts qui inclut des relations de similarité supplémentaire telles que : « est un genre de », « est un exemple spécifique de », « est une partie de », « est le contraire de », …

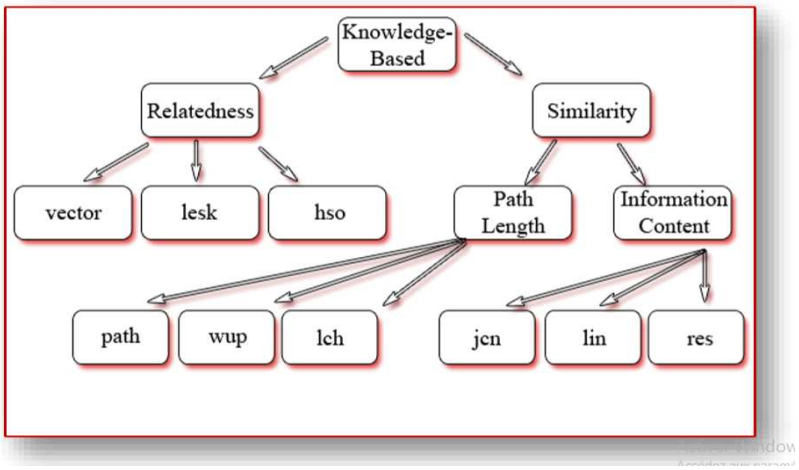

Figure 6 Quelques mesures de similarité basée connaissance

2.3. Les approches hybrides

Les méthodes hybrides utilisent des mesures de similarité multiples, soit en combinant des méthodes sémantiques et syntaxiques à la fois où bien plusieurs méthodes de même type. De nombreuses recherches ont couvert ce domaine et ont démontré qu'une telle approche est nettement plus performante que les approches individuelles. On peut citer pour l'exemple l'expérience de Mihalcea [13] où huit mesures de similarité sémantique ont été testées, deux de ces mesures étaient des mesures fondées sur des corpus et les six autres étaient basées sur des connaissances. Premièrement, ces huit algorithmes ont été évalués séparément, puis ils ont été combinés ensemble. La meilleure performance a été obtenue en utilisant une méthode qui combine plusieurs métriques de similarité en une seule. Une autre recherche concernant les approches hybrides est celle Islam & al. [14] qui ont présenté une méthode et l'ont appelée similitude de texte sémantique (STS). Cette méthode détermine la similarité de deux textes grâce à une combinaison entre des informations sémantiques et syntaxiques. Ils ont considéré deux fonctions obligatoires (similarité de chaîne et similarité de mots sémantique) et une fonction optionnelle (similarité d'ordre de mots communs). La méthode STS a obtenu un très bon coefficient de corrélation de Pearson pour 30 paires de données

3. Les Word Embedding

3.1. Définitions et généralités

Le Word Embedding [15] désigne un ensemble de techniques de machine learning qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels. Cette nouvelle représentation a ceci de particulier que les mots apparaissant dans des contextes similaires possèdent des vecteurs correspondants qui sont relativement proches. Par exemple, on pourrait s'attendre à ce que les mots « chien » et « chat » soient représentés par des vecteurs relativement peu distants dans l'espace vectoriel où sont définis ces vecteurs. Cette technique est basée sur l'hypothèse (dite « de Harris » ou « distributional hypothesis ») qui veut que les mots apparaissant dans des contextes similaires ont des significations apparentées. La technique des word embeddings diminue la dimension de la représentation des mots en comparaison d'un modèle vectoriel par exemple, facilitant ainsi les tâches d'apprentissage impliquant ces mots, puisque moins soumis au fléau de la dimension

Les WE caractérisent chaque mot par un ou plusieurs vecteurs denses, de faible dimension ayant des éléments réels, capturant les spécificités latente (de contexte) du mot et les propriétés syntaxiques et sémantiques utiles.

Il existe plusieurs approches de word embedding. Les premières remontent aux années 1960 et reposent sur des méthodes de réduction de dimensionnalité. Plus récemment, de nouvelles techniques basées sur des modèles probabilistes et des réseaux de neurones, comme Word2Vec, ont permis d'obtenir de meilleures performances.

Word2Vec [16] est un modèle de représentation distribué des mots peu profond (par rapport à un réseau de neurone traditionnel) où le principe est de générer des vecteurs dimensionnels à partir d'un apprentissage sur des données d'entrée non étiquetées. En fait, il prédit les mots en fonction de leur contexte en utilisant l'un des deux modèles neuronaux distincts : CBOW et Skip-Gram.

3.2. Continuous Bag-of-Words « CBOW »

Le modèle « CBOW » prédit un mot courant basé sur son contexte. Le contexte correspond à un certain nombre de mots voisins à gauche et à droite du mot. Dans le processus CBOW, trois couches sont utilisées comme nous pouvons le voir sur la figure 7. La couche d'entrée

correspond au contexte. La couche cachée correspond à la projection de chaque mot de la couche d'entrée dans la matrice de poids qui est projetée dans la troisième couche qui est la couche de sortie. La dernière étape de ce modèle est la comparaison entre sa sortie et le mot lui-même afin de corriger sa représentation en fonction de la propagation arrière du gradient d'erreur. Ainsi, le but du réseau de neurones CBOW est de maximiser l'équation suivante [17] :

Où V (et C) est la taille du vocabulaire (contexte).

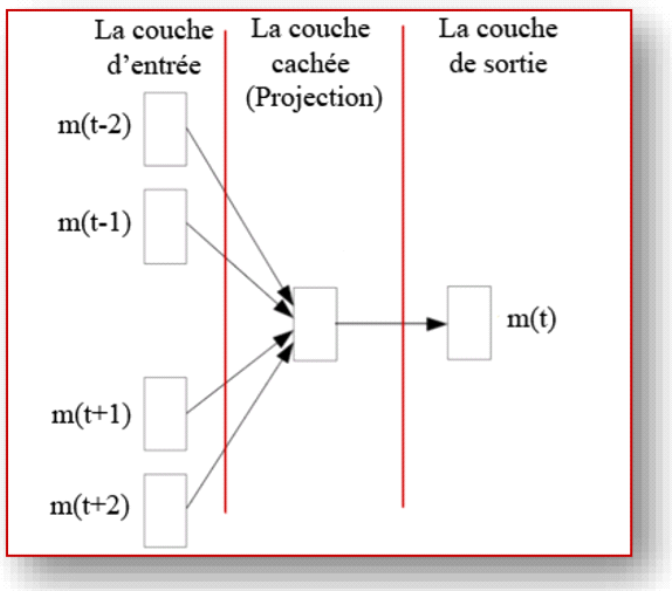

Figure 7 Schéma du modèle CBOW [15]

3.3. Skip-Gram « SG »

Skip-Gram : c'est le contraire des modèles CBOW. En effet, la couche d'entrée correspond au mot cible et la couche de sortie correspond au contexte comme on peut le voir sur la figure 8. Ainsi, Skip-Gram cherche la prédiction du contexte d'un mot donné au lieu de la prédiction d'un mot sachant son contexte comme CBOW. La dernière étape de Skip-Gram est la comparaison entre sa sortie et chaque mot du contexte afin de corriger sa représentation en fonction de la propagation arrière du gradient d'erreur. En fait, il cherche la maximisation de l'équation suivante [17]:

$$\frac{1}{V}\sum_{t=1}^{V}\sum_{j=t-c,j\neq t}^{t+c}\log p(m_j|m_t)$$

Où V (et C) est la taille du vocabulaire (contexte).

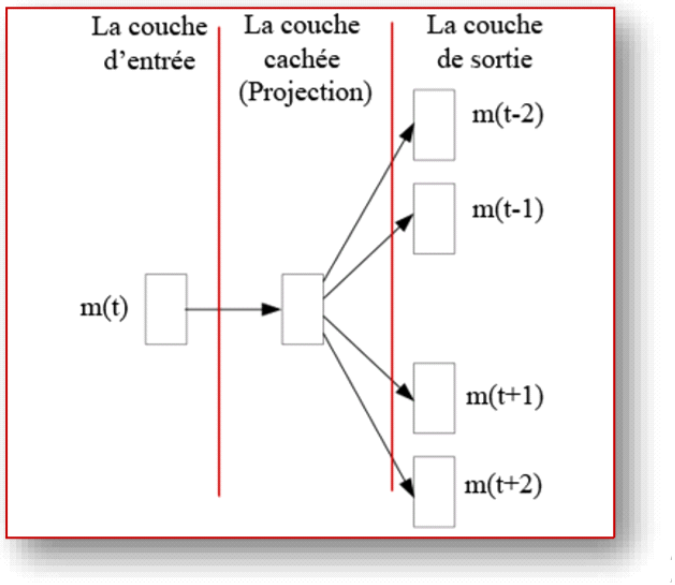

Figure 8 Schéma du modèle Skip-Gram [15]

Chacun de ces modèles à son propre avantage. À titre d'exemple, Skip-Gram est plus efficace avec de petites données d'entraînement. De plus, les mots peu fréquents sont bien présentés. D'un autre côté, CBOW fonctionne bien avec des mots fréquents.

4. Revue sur quelques travaux connexe dans le domaine des ASAG

De nombreux chercheurs s'intéressent au thème de l'évaluation automatique des réponses courtes et ainsi qu'à la similarité. Chaque approche à ses caractéristiques : langue, domaine, ressources nécessaires.

4.1. Les travaux sur la similarité de textes utilisant la langue anglaise

1) Concernant les similarités syntaxiques :

Plusieurs mesures de similarité et hybridations on été employé dans différents projet, notamment les mesures de similarité syntaxique, dans un projet [20], la mesure de similarité pondérée Jaccard a été utilisée. Pour comparer les contextes syntaxiques de deux mots, ils ont utilisé comme mesure de similarité une version pondérée de la mesure de Jaccard binaire. La mesure de Jaccard binaire, notée BJ, calcule la valeur de similarité entre deux mots, . et /, en comparant les attributs qu'ils partagent et ne partagent pas

D'autres ont utilisé en outre un noyau d'arbre partiel [19] pour calculer la similarité entre la question et le commentaire en fonction de leur correspondance arbres syntaxiques peu profonds. Ces arbres ont le mot lemmata sous forme de feuilles, puis il existe un parent de nœud de balise POS (Part Of Speech) pour chaque feuille de lemme, et les nœuds de balise POS sont dans des tours groupés sous morceaux d'analyse superficielle, qui sont liés à un nœud de phrase racine; enfin, tout root

Les nœuds de phrases sont liés à une super-racine pour toutes les phrases de la question / commentaire.

2) Similarités sémantiques

Des linguistes ont proposé des méthodes pour mesurer la similarité. Une des œuvres pionnières est : ' sémantique différentiel '[20] qui analyse le sens des mots dans une gamme de dimensions différentes avec les adjectifs opposés aux deux extrémités, et localise les mots dans la sémantique espace.

Les travaux récents sur la représentation des connaissances sont quelque peu liée au différentiel sémantique d'Osgood [20]. La plupart d'entre eux décrivent le sens des mots en utilisant des symboles comme des microfeatures [21] qui correspondent aux dimensions sémantiques.

Cependant, les problèmes suivants découlent de la procédure différentielle sémantique comme

mesure du sens. La procédure ne repose pas sur le sens dénotatif d'un mot, mais uniquement sur les émotions connotatives attachées au mot; c'est difficile de choisir les dimensions pertinentes, c.-à-d. les dimensions requis pour l'espace sémantique suffisant.

D'autres ont appliqués trois approches pour construire des représentations vectorielles contenant des mots, en utilisant

(i) L'analyse sémantique latente [22], formé sur le corpus Qatar Living avec un mot fenêtre de cooccurrence de taille ± 3 et produisant un vecteur de 250 dimensions avec SVD (ils ont produit un vecteur pour chaque nom du vocabulaire);

(ii) GloVe [23], à l'aide d'un modèle préformé sur Common Crawl (42B jetons), avec 300 dimensions;

(iii) COMPOSES [24], en utilisant des vecteurs prédictifs de 400 dimensions estimés précédemment. Ils ont représentés à la fois q et c comme une somme des vecteurs correspondant aux mots en leur sein (en négligeant le sujet de c). Ils ont calculés la similarité cosinus pour estimer sim (q, c).

Ils ont également expérimenté Word2vec [25] vecteurs préformés à la fois avec arc et skipgram sur les données d'actualités et aussi avec word2vec et les vecteurs GloVe formés sur les données de Qatar Living, mais ils les ont jetés car ils ne les ont pas aidés au-dessus de toutes les autres fonctionnalités que ils avaient.

4.2. Les travaux connexes à notre recherche

Les travaux que nous menons dans le cadre d'une approche hybride qui permet de combiner plusieurs approches syntaxiques et sémantiques (particulièrement basés sur le corpus) est celle basées sur les Word Embedding. Dans ce contexte notre travail est connexe aux travaux de [26] qui présente un système de notation automatisé qui note les tests d'écriture en anglais des étudiants. Le système fournit un score et une rétroaction de diagnostic aux étudiants sans effort humain. Les utilisateurs ciblés sont les étudiants coréens des collèges qui apprennent l'anglais comme langue seconde. Le système prend une seule phrase anglaise en entrée. Traiter avec une seule phrase en tant qu'entrée à certains avantages sur la comparaison de l'entrée avec les réponses données par des enseignants humains et en donnant une rétroaction détaillée aux étudiants. Le système a été développé et testé avec les données de test réelles recueillies lors de tests d'anglais donnés à des étudiants de troisième année du premier cycle du secondaire. La

notation nécessite deux étapes du processus. Le premier processus analyse la phrase d'entrée afin de détecter les erreurs possibles, telles que les erreurs d'orthographe et les erreurs de syntaxe. Le deuxième processus consiste à comparer la phrase d'entrée avec les réponses données pour identifier les différences comme des erreurs. Pour évaluer les performances du système, la sortie produite par le système est comparée au résultat fourni par des évaluateurs humains. La concordance de score entre un évaluateur humain et le système est assez proche de celle entre deux évaluateurs humains.

5. Principes et méthodes du système traitant l'arabe

Nous allons expliquer le principe de ce système [27], puis le fonctionnement de ses approches, vu que notre travail consiste en une adaptation de ce même système a l'anglais :

Ce système suit le même pipeline de développement de systèmes ASAG (figure 3), il considère un mécanisme basé sur le Stemming afin d'analyser l'impact sur l'évaluation automatique des questions à réponses courtes en langue arabe. Les techniques de stemming ont été exploitées en combinaison avec les mesures de similarités. Un algorithme de stemming peut être défini comme la procédure de réduction de tous les mots qui partagent la même racine à une forme commune [28].

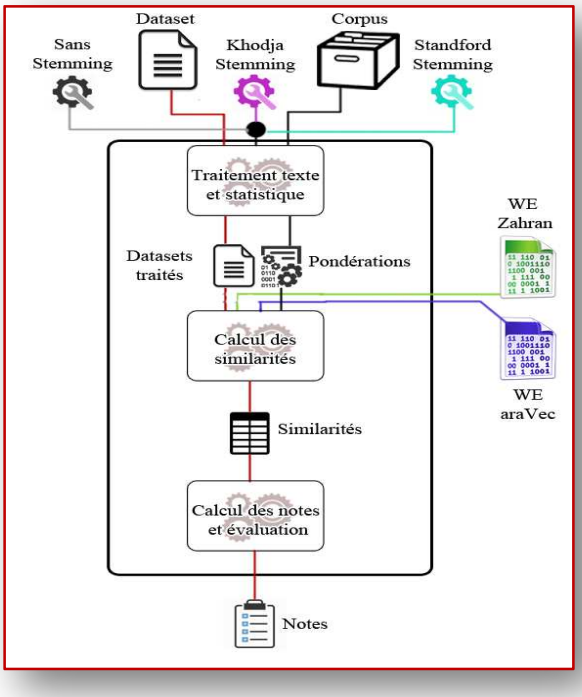

Figure 9 Approche méthodologique dans le développement du système [27]

Pour toutes les approches de similarités développées ils ont considéré les 3 cas suivants représentés dans la figure 9 :

1. Aucune technique de stemming n'est considérée aux deux réponses (Réponse de l'étudiant et la réponse Modèle) à comparer et qui sont considérées dans leur nature brute. Cette technique est représentée par la couleur noire sur le schéma.
2. Une technique de stemming lourde (Heavy Stemming) est appliquée aux réponses à comparer. Le streaming lourd, également appelé « Root-Stemming » (Stemming à la racine), consiste à supprimer les préfixes et les suffixes bien connus pour extraire la racine réelle d'un mot et à identifier le motif en correspondance avec le mot restant. La couleur violète illustre cette méthode.
3. Une technique de stemming légère (Light Stemming) est appliquée aux réponses à comparer. Le streaming léger est un processus moins complexe, où le stemming est arrêté

sur la suppression des préfixes et des suffixes, sans tenter d'identifier la racine réelle du mot. Cette dernière technique est identifiée par la couleur bleue-claire.

Il est utile de noter qu'hormis les effets de l'approche appliquée sur les données, le processus est exactement identique pour les 3 approches. Aussi, les méthodes de calcul de similarité qui ont été utilisées nécessitent des ressources telles que les WEs ou les corpus qu'il faut acquérir avant de faire recours à ces dernières.

Plusieurs méthodes de calcul de similarité font recours aux méthodes statistiques pour améliorer les résultats de similarité en accordant de l'importance à la fréquence des mots dans les corpus ainsi qu'à l'étiquetage morpho syntaxique, Pour le calcul de fréquences, les différentes pondérations appliquées sont détaillées dans le chapitre trois:

- IDF
- TF-idf
- TFminMax
- TFlog

Un étiquetage morphosyntaxique a également été appliqué a l'aide de l'outil Stanford Tagging

- **Calcul des similarités : Pour chaque approche, tous les modules de calcul de similarité sont exécutés avec les ressources adéquates à l'approche.**

- **Calcul des similarités**

Le calcul de similarité entre les réponses modèles et les réponses des apprenants est le noyau du travail. Pour chaque approche, tous les modules de calcul de similarité sont exécutés avec les ressources adéquates à l'approche. Le module « Calcul des similarités » est un regroupement de plusieurs sous module de calcul des similarités indépendants les uns des autres qu'ils ont utilisé. Notons que, dans la continuité de cette approche méthodologique, les datasets ont été traités selon une approche de stemming, de ce fait, les résultats renvoyés par les modules de calculs sont dépendants de l'approche appliquée.

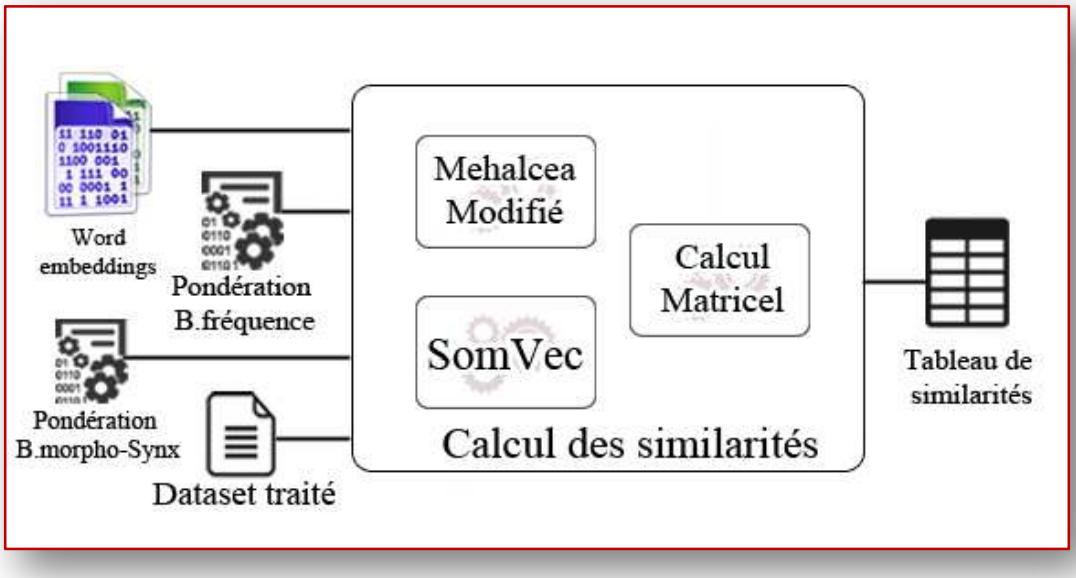

Figure 10 Composants du module « Calcul des similarités »

- **Calcul des scores et évaluation, cette tâche est affectée au module portant le même nom « calcul de score et évaluation » les détails serons présenté dans le chapitre deux.**

Conclusion

Nous avons présenté quelques domaines et approches chevauchant notre projet ainsi que les définitions nécessaires pour la compréhension de la suite, mais également les principes du système traitant l'arabe qu'on va adapter pour la langue anglaise au cours de ce projet.

Dans le prochain chapitre, nous allons détailler d'avantage le fonctionnement du système ainsi que les différents outils nécessaires à l'adaptation pour l'anglais.

Chapitre 2

Adaptation du système d'évaluation automatique à l'anglais

Dans ce chapitre, nous allons détailler les différentes méthodes statistiques mais également les différentes méthodes de calcul de similarités utilisé, ainsi que toutes les ressources et les outils traitant la langue anglaise acquise et utilisé dans ce travail

1.Acquisition des ressources :

1.1. L'acquisition des Word Embeddings

Les word embeddings sont générés grâce à des réseaux de neurones, ce processus fait trainer des corpus de milliards de mots, ce qui nécessite du matériel très performant. En plus, une bonne représentation dépend de beaucoup de paramètres comme nous allons le voir par la suite, ce qui fait qu'énormément de tests doivent être effectués pour aboutir à des vecteurs de qualité. L'objectif dans ce travail étant d'évaluer l'impact des Word Embeddings sur le processus d'évaluation en langue anglaise, nous avons utilisé deux modèles de WE déjà générés pour les mots de la langue anglaise décrits dans [15]

*** Les Word Embedding de Word2vec[1]** [29] **:**

L'outil word2vec prend un corpus de texte en entrée et génère les vecteurs de mots en sortie. Il construit d'abord un vocabulaire à partir des données de texte d'apprentissage, puis apprend la représentation vectorielle des mots. Le fichier de mot vecteur résultant peut être utilisé comme caractéristique dans de nombreuses applications de traitement de langage naturel et d'apprentissage automatique.

Word2Vec est une solution efficace à ces problèmes, qui exploite le contexte des mots cibles. Essentiellement, nous souhaitons utiliser les mots environnants pour représenter les mots cibles avec un réseau de neurones dont la couche cachée code la représentation des mots.

Il existe deux types de Word2Vec, de saut de gramme (Skip-gram) et de sac de mots continu (CBOW).

On dit que Skip-gram a tendance à faire mieux avec des mots rares. Néanmoins, les performances de Skip-gram et de CBOW sont généralement similaires.

Nous avons utilisé des vecteurs préformés formés sur une partie de l'ensemble de données Google News (environ 100 milliards de mots). Le modèle contient des vecteurs de 300 dimensions pour 3 millions de mots et de phrases. Les phrases ont été obtenues en utilisant une

[1]Word2vec: https://code.google.com/archive/p/word2vec/

approche simple basée sur les données décrites dans [28]

***FastText** [29][2]

FastText est une extension de Word2Vec proposée par Facebook en 2016. Au lieu d'insérer des mots individuels dans le réseau de neurones, FastText divise les mots en plusieurs n-grammes (sous-mots). Par exemple, les trigrammes du mot ' apple ' sont 'app', 'ppl' et 'ple' (en ignorant le début et la fin des limites de mots). Le mot vecteur d'incorporation pour pomme sera la somme de tous ces n-grammes. Après la formation du réseau de neurones, nous aurons des intégrations de mots pour tous les n-grammes en fonction du jeu de données de formation. Les mots rares peuvent maintenant être correctement représentés, car il est fort probable que certains de leurs n-grammes apparaissent également en d'autres termes.

Bien que le modèle FastText nécessite plus de temps (nombre de n-grammes> nombre de mots), il fonctionne mieux que Word2Vec et permet aux mots rares d'être représentés de manière appropriée.

Quelques caractéristiques des WE utilisé :

WE	taille	Dimensions	Corpus de génération
Google news CBOW	3,39 Go	300	Google news
FastText CBOW	6,73 Go	300	Common Crawl et Wikipédia
FastText Skip-gram	7,91 Go	300	Wikipedia

Tableau 1 Caractéristiques des WE utilisés

1.2. Acquisition des corpus

Dans l'optique l'optimiser les calculs de similarité, nous avons utilisé des modèles statistiques expliqués plus loin dans ce chapitre. Ces modèles nécessitent, comme ressources, de grands corpus contenant des milliards de mots. Notre travail portant sur la langue anglaise, nous avons parcouru la littérature à la recherche de corpus anglais volumineux et représentatifs et surtout traitable par nos machines. Les corpus les plus cités et libres de droit que nous avons utilisé sont

[22] Fasttext : https://fasttext.cc/

: BBC [32], [3] Gutenberg [33], voici leurs caractéristiques :

	BBC English	Gutenberg
Nombre de mots total	486 241	210 913 418
Nombre de mots uniques	27 757	2 922 859
Nombre de documents	2 225	3 037
Contenus	-Divertissement -Businesse -Politique -Sport -Technologie	-Culture -Electronique -Religion -Economie -Actualités local -Actualité internationale -Sports

Tableau 2 Caractéristiques des corpus utilisés

Le module « Traitement des textes et statistique » est un élément clé qui assure le formatage des données selon les critères de cette dernière, la cohérence entre le traitement des ressources et l'estimation de l'importance des mots dans la langue anglaise. Il comporte 3 sous modules chargés de : la normalisation, le Stemming et le calcul de fréquence et l'étiquetage morphosyntaxique (POS).

1.3. Les jeux de données (Datasets)

Afin d'évaluer les méthodes de notation des réponses courtes, nous avons utilisés un ensemble de données de questions provenant de travaux d'initiation à l'informatique, avec des réponses fournies par une classe d'étudiants de premier cycle. Les tâches ont été administrées dans le

[3] Gutenberg : https://web.eecs.umich.edu/~lahiri/gutenberg_dataset.html
BBC : http://mlg.ucd.ie/datasets/bbc.html

cadre d'un cours sur les structures de données à l'Université de North Texas [32]. Pour chaque devoir, les réponses des étudiants ont été collectées via l'environnement d'apprentissage en ligne WebCT.

Les évaluations rapportées dans le présent document portent sur les réponses soumises pour trois des travaux de cette classe. Chaque tâche consistait en sept questions à réponse courte.1 Trente étudiants ont été inscrits dans la classe et ont soumis des réponses à ces tâches. Ainsi, l'ensemble de données avec lequel nous travaillons comprend un total de 609 réponses d'élèves (3 travaux x 7 questions / devoir x 29 réponses d'élèves / question). Les réponses ont été notées indépendamment par deux juges humains, en utilisant une échelle entière allant de 0 (complètement incorrect) à 5 (réponse parfaite). Les deux juges humains étaient des étudiants diplômés en informatique ; l'un était l'assistant d'enseignement de la classe Structures de données, l'autre était l'un des auteurs de cet article. Le tableau 2 montre deux paires de questions avec trois exemples de réponses d'élèves. Les notes attribuées par les deux juges humains sont également incluses.

	Question: What is the role of a prototype program in problem solving?
	Réponse : **To simulate the behaviour of portions of the desired software product.**
Notes	*Réponses des étudiants*
5	A program that simulates the behavior of portions of the desired software product.
2	To lay out the basics and give you a starting point in the actual problem solving.
4,5	To simulate problem solving for parts of the problem
	Question: What are the main advantages associated with object-oriented programming?
	Réponse : **Abstraction and reusability.**
Notes	*Réponses des étudiants*
2	Using different modules allows for easier debugging
4	Existing classes can be reused Program maintenance and verifications are easy
3,5	Data encapsulation concept,the use of functions or methods to manipulate data.

Tableau 3 Deux exemples de questions avec réponses courtes du dataset de Molher fournies par les étudiants et les notes attribuées par les deux juges humains

Les évaluations sont effectuées à l'aide du coefficient de corrélation de Pearson, mesuré par rapport à la moyenne des notes attribuées par l'homme, question par question et par mission.

Dans la configuration par question, chaque question et la réponse correspondante de l'élève sont considérées comme un point de données indépendant dans la corrélation ; l'accent est donc mis sur la correction de la note attribuée à chaque réponse.

1.4. La normalisation

Toute application traitant des documents textuels comporte un module de prétraitement de ces derniers. Cette étape permet la prise en considération des contraintes liées à la langue. Outre, il est nécessaire de prétraiter les données brutes afin de pouvoir ensuite les traiter avec des processus unifiés et non une multitude de processus adaptés à tous les cas possibles.
La normalisation vise à unifier les diverses manières d'écrire un même mot, à corriger les fautes ou les incohérences évidentes. Etant donné que nous allons chercher la représentation des mots dans les modèles de Word2vec et FastText, le texte des réponses doit subir le même traitement qu'ont subi les données d'entrainement des modèles :

- Suppression des nombres des deux réponses
- Suppression des signes et des symboles
- Suppression de toutes les lettres d'autres langues.

1.5. Acquisition des Stemmers

Un algorithme de stemming peut être défini comme la procédure de réduction de tous les mots qui partagent la même racine à une forme commune. Le stemming consiste dans notre cas à réaliser les actions suivantes pour chaque couple de réponses à comparer. :

- Supprimer le préfixe
- Supprimer le suffixe.
- Supprimer les mots vides.

Les mots d'arrêt ou les stopwords sont des mots considérés sans importance pour le calcul de similarité et sont généralement supprimés des traitements (aussi bien dans les réponses que dans les corpus) car ils renvoient une grande quantité d'informations inutiles. Ces derniers peuvent être calculés à partir d'un seuil de fréquence sur un corpus assez volumineux, néanmoins il existe plusieurs listes de stopwords plus ou moins similaires qui sont déjà calculées et rendues disponibles sur le net. Chaque langage de programmation donnera sa propre liste de

mots vides à utiliser. Ce sont principalement des mots couramment utilisés en anglais, tels que : ' as, the, be, are, … etc '.

Pour notre cas, nous avons maintenu la liste par défaut fournie par l'outil de stemming que nous avons utilisé.

La principale différence entre l'algorithmes de porter et de lancaster est que lancaster est nettement plus agressif que porter. Les trois principaux algorithmes d'extraction utilisés de nos jours sont Porter, Snowball (Porter2) et Lancaster (Paice-Husk), le continuum d'agressivité allant dans le même sens. Porter est l'algorithme le moins agressif.

Porte[4]r [35]**:** étant le plus ancien développé à l'origine en 1979 Le stemmer le plus couramment utilisé est sans aucun doute l'un des plus doux. L'un des rares serveurs à prendre en charge le support Java, bien que ce soit également l'algorithme le plus intensif en calculs (octroyé de façon peu significative), il ne suit pas la linguistique, mais plutôt un ensemble de 05 règles pour différents cas qui sont appliquées par phases (étape par étape).) pour générer des stems, C'est la raison pour laquelle PorterStemmer ne génère pas souvent de stem qui sont de vrais mots anglais. Il ne conserve pas de table de recherche pour les stems réelles du mot mais applique des règles algorithmiques pour générer des stems. Il utilise les règles pour décider s'il est sage de supprimer un suffixe. On peut générer son propre ensemble de règles pour n'importe quel langage, c'est pourquoi Python a introduit Snowball Stemmers les méthodes utilisées pour créer des Stemmers non anglais

Alors pourquoi l'utiliser ? PorterStemmer est connu pour sa simplicité et sa rapidité. Il est généralement utile dans les environnements de récupération d'informations**,** connus sous le nom d'Environnements IR, pour un rappel et une extraction rapide des requêtes de recherche. Dans une RI typique, les documents d'environnement sont représentés comme des vecteurs de mots ou de termes. Les mots ayant la même racine auront une signification similaire

Porter 2 ou lovins (Snowball) [34] **:** est[5] une extension de Porter, il a été créé par Martin Porter, presque universellement considéré comme une amélioration par rapport à porter, et pour cause. Porter lui-même admet en fait qu'il est meilleur que son algorithme d'origine.

[4] Porter : http://snowball.tartarus.org/algorithms/porter/stemmer.html

[5] SnowballStem : https://snowballstem.org/

Lancaster : https://www.nltk.org/_modules/nltk/stem/lancaster.html

Temps de calcul légèrement plus rapide que celui de Porter, avec une communauté assez large autour.

Lancaster (Paice-Husk) [37]**:** est le plus agressif, c'est un algorithme itératif avec les règles enregistrées à l'extérieur. Une table contenant environ 120 règles indexées par la dernière lettre d'un suffixe. À chaque itération, il essaie de trouver une règle applicable par le dernier caractère du mot. Chaque règle spécifie une suppression ou un remplacement d'une fin. Si cette règle n'existe pas, elle prend fin. Il se termine également si un mot commence par une voyelle et qu'il ne reste que deux lettres ou si un mot commence par une consonne et qu'il ne reste que trois caractères. Sinon, la règle est appliquée et le processus se répète.
LancasterStemmer est simple, mais il peut en résulter des surtensions lourdes dues aux itérations. Le surmenage fait que les stems ne sont pas linguistiques ou peuvent n'avoir aucune signification.

Ce dernier étant un stemmer plus lourd peut parfois s'éloigner de la signification du mots, chose qui ne contribue pas a l'amélioration des résultats, nous avons donc choisi d'utiliser les deux stemmers suivants pour la suite du travail :

- Porter Stemmer pour un stemming léger
- Snowball Stemmer pour un stemming un peu plus lourd

2. Fonctionnement des modules et modèles de similarité du système utilisé

2.1. Calcul de fréquence et étiquetage morphosyntaxique

- **Pondération des mots basée sur la fréquence de mots dans le corpus**

Une des mesures les plus utilisés pour estimer l'importance des mots est la pondération IDF (Inverse Document Frequency) . Elle opère une transformation des effectifs bruts des mots qui sont calculés dans le cadre du traitement de texte, et permet d'exprimer simultanément les

fréquences auxquelles certains termes ou mots spécifiques apparaissent dans un ensemble de documents, ainsi que leurs spécificités sémantiques. Le principe de l'approche est de déterminer les mots les plus pertinents dans un texte et ceux qui sont insignifiants comme les stopwords. Ainsi les mots les moins fréquents sont les plus discriminants.

Cette mesure est généralement utilisée avec le calcul des TFs (Term Frequency) qui consiste à calculer la fréquence d'un mot par rapport à un document. Plusieurs variations de TF existent comme nous pouvons le voir dans la figure 10 :

Schéma de pondération	formule du TF
binaire	$0, 1$
fréquence brute	$f_{t,d}$
normalisation logarithmique	$1 + \log(f_{t,d})$
normalisation « 0.5 » par le max	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
normalisation par le max	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Figure 11 Variantes de TF [36]

Le **TF-IDF** est une mesure largement utilisé, elle permet d'avoir une estimation globale sur l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus, elle est obtenue en multipliant **TF** par **IDF** :

TF-IDF (W) = IDF (W) * TF (W)

Dans le cadre de notre travail, les documents sont substitués par les réponses à comparer. Une réponse est de toute évidence beaucoup plus petite par rapport à un document, ce qui fait que les propriétés sémantiques des mots la composant sont nettement moins explicites et difficile à cerner. Pour cette raison, ils ont proposé dans cette approche d'autres façons de pondérer en considérant les mots par rapport à un corpus représentatif et par conséquent, à la langue de façon générale :

- **La pondération TFminmax** [39]**:**

La pondération TFminmax utilise une autre mesure statistique calculée par le logarithme du nombre de fois où le mot apparait dans le corpus divisé par le nombre de mot total dans le corpus :

$$\text{TFlog (W)} = -\log\left(\frac{Cpt}{N}\right)$$

Où Cpt et le nombre de fois qu'apparait W dans le corpus, N = nombre de mots total dans le corpus. Une fois les TFlogs de tous les mots calculés, une normalisation est appliquée à ces derniers afin d'obtenir les TFminmax en divisons le tf-log du mot par le max des tf-logs obtenus :

$$\text{TFminmax(W)} = \frac{\text{TFlog(W)}}{Max(\text{TFlogs})}$$

- **La pondération TF-IDFminmax :**

Contrairement à TF-IDF de base que nous avons vue précédemment et où le but est d'avoir une estimation globale sur l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus, le TF-IDFminmax génère une estimation globale de l'importance du mot par rapport à la langue entière. Cette estimation est obtenue par la formule :

TF-IDFminmax(W)= IDF(W) * TFminmax (W)

- Pondération des mots basée sur l'étiquetage morpho-syntaxique POS (Part Of Speech) :

Contrairement à la pondération de mots basée sur la fréquence des termes qui tire ses principes des méthodes statistiques, la pondération basée sur l'étiquetage morphosyntaxique fait recours au domaine linguistique. L'avantage principal de cette approche est qu'elle est indépendante et qu'elle ne nécessite pas des corpus ou de ressource externe à l'analyseur morphosyntaxique. C'est d'ailleurs pour cette raison que nous nous sommes orientés vers cette approche. Pour identifier les mots les plus importants dans un texte, cette approche se base sur les caractéristiques grammaticales et syntaxiques de la langue. D'abord un analyseur

morphosyntaxique opère une analyse des textes afin de pouvoir classer les mots dans leurs catégories (Nom, verbe, adjectifs…). Ensuite, selon les propriétés de la langue utilisée, chaque catégorie se voit attribuer une valeur représentant son importance dans la langue.

Dans notre cas, nous avons utilisé l'analyseur POS Tagger de NLTK Python pour générer l'étiquetage des réponses, ensuite, nous avons réalisé une série de tests qui a abouti à la considération de 4 catégories : les noms "N", les verbes "V", les adjectifs "AJ" et la 4eme catégories "A" pour le reste.

2.2. Mesures de similarité utilisée

Nous avons utilisé les 3 mesures de similarités développées dans le travail précèdent traitant la langue arabe qui sont :

- SOMVEC
- Mehalcea modifiée
- Matrices d'ordre

Pour chacune de ces méthodes, nous avons calculé les pondérations des corpus de textes avec les différentes méthodes de calcules de fréquences, et ainsi ayant obtenue plusieurs approches de mesures de similarité pondéré pour chacune des 3 méthodes principale, voici donc les détails de ces méthodes :

Le calcul de similarité des systèmes se fait au niveau des phrases pour certains et au niveau des mots pour d'autres :

A- Méthodes basées sur les similarités de phrases

Les méthodes basées sur la similarité au niveau des phrases appliquent un principe relativement simple : elles traitent la réponse modèle indépendamment de la réponse de l'apprenant dans un premier temps, puis, une fois les deux phrases représentées d'une façon adéquate, on fait appel à une mesure de similarité pour évaluer la similitude entre les deux représentations. Le module SomVec implémente une méthode basée sur la similarité de phrase :

A.1- La somme des vecteurs : avec et sans pondération [40]

La somme des vecteurs est largement utilisée dans la littérature, elle consiste à sommer les vecteurs représentatifs de chaque mot de la phrase avant de calculer la similarité cosinus entre la somme des deux phrases. Soit :

- Une phrase S1 composée des mots M_1, M_2 … M_N
- Une phrase S2 composée des mots K_1, K_2 … K_N

La première étape est de calculer V1 et V2 :

- V1 = $\sum_{i=1}^{N} v(M_i) * \beta$
- V2= $\sum_{i=1}^{N} v(K_i) * \beta$

Ensuite il suffit de calculer la similarité cosinus entre les deux vecteurs :

- simCos(V1,V2) = cos (V1,V2)

Dans la somme des vecteurs simple, aussi appelé somme des vecteurs unitaire, le coefficient β est égal à 1. Nous considérons cette configuration comme Baseline du module d'évaluation comme nous allons le voir par la suite dans le chapitre 4.

Une amélioration de cette méthode consiste à modifier la pondération des vecteurs de mot en affectant à β une valeur représentative de l'importance de chaque terme. Sur ce principe, nous avons étendu le fonctionnement du module SOMVEC afin de pondérer suivant les approches de pondérations implémentées par le module « calcul des fréquences et étiquetage » qui sont : IDF, TFminmax, TFlog, TF-IDFminmax, POS, MIXTE (combinaison entre TFlog et POS).

B- Méthodes basées sur les similarités de mots

Les approches de calcul de similarité basée sur les mots consistent à calculer la similarité de chaque mot d'une phrase avec tous les des mots de l'autre phrase, avant de passer vers la similarité des globale entre ces phrases suivant une formule qui défère d'une méthode à une autre. Nous allons détailler ici les méthodes que nous avons implémentées :

B.1- Calcul matriciel mot à mot (ou Matrice d'ordre) :

Le module « calcul matriciel » est implémenté par cette méthode. Cette approche inspirée des travaux de [14] utilise deux fonctions afin de calculer la similarité entre deux phrases : une fonction optionnelle de similarité d'ordre des mots en communs pour incorporer des informations syntaxiques dans le calcul, et une méthode de calcul de similarité sémantique mot à mot en utilisant les Word Embeddings.

Pour aller plus en profondeur, détaillons le fonctionnement des deux fonctions :

1) Similarité d'ordre des mots en commun dans les deux réponses

Cette similarité donne une importance aux mots en commun suivant leurs ordres dans les phrases à comparer, Pour deux phrases P et R de taille m et n (m étant inférieure ou égale n sinon nous inversons les deux phrases), Nous retirons les K mots en commun de chaque phrase P et R, et nous les mettons dans les vecteurs X et Y respectivement en suivant le même ordre dans lequel ils apparaissaient dans P et R.

La 2eme consiste à remplacer chaque mot dans X par sa position dans X, en démarrant de 1 jusqu'à arriver à α = |X|, ensuite remplacer dans Y chaque terme Yi où : Xi=Yi par le numéro remplacé avec ce dernier dans X. Ceci fait, le prochain travail à faire est de calculerS_0.

$$S0 = \begin{cases} 1 - \dfrac{2\sum_{i=1}^{\alpha} |X_i - Y|}{\alpha^2} & Si\ \alpha\ \text{est pair} \\ 1 - \dfrac{2\sum_{i=1}^{\alpha} |X_i - Y|}{\alpha^2 - 1} & Si\ \alpha\ \text{est impair et } \alpha > 1 \\ 1 \end{cases}$$

2) Similarité sémantique mot par mot :

C'est une procédure itérative qui intervient après avoir retrié les mots en communs des phrases P et R pour construire la matrice comme suite :

D'abord, Il faut construire une matrice de taille (m-α)*(n-α) de sorte que la ligne contient la phrase la plus courte, sinon nous inversons entre la ligne et la colonne puis on calcule la similarité entre chaque mot Pi et Rj en appliquant le cosinus sur leurs vecteurs : Cosinus (V(Pi), V(Rj)).

Une amélioration possible est d'appliquer une pondération sur la matrice d'ordre, il suffit de multiplier les vecteur V(Pi) et V(Ri) par la pondération voulu avant de calculer le cosinus. Nous avons étendu ce module calcul matriciel pour qu'il prenne en charge la pondération MIXTE (TFminmax et POS).

Une fois la matrice construite, il faut extraire le maximum de la matrice pour l'ajouter à une liste de max (qu'on notera Max i pour la suite) et retirer la ligne et la colonne de ce maximum.

Ces étapes doivent être répétées jusqu'à la satisfaction d'une des deux conditions :

- La somme des éléments de la matrice devient nulle
- Ou bien : m – α – |P|<=0

Une fois une des deux conditions atteintes, la similarité entre P et R peut être calculé comme suit :

$$\boldsymbol{S(P, R) = \frac{(\alpha(1 - W_f + W_f S_0) + \sum_{i=1}^{|Maxi|} Maxi_i) + (m+n)}{2mn}}$$

Wf est la pondération accordée à l'ordre pour les mots en communs, Nous pouvons ignorer l'ordre en affectant un 0 à Wf ce qui revient à calculer la formule suivante :

$$S\,(Phrase1, Phrase\ 2) = \frac{(\alpha + \sum_{i=1}^{|Maxi|} Maxi_i) + (m+n)}{2mn}$$

B.2- L'approche de Mihalcea modifiée [13]**:**

Cette approche a été implémenté par le module « Mihalcea Modifié ». Chaque mot de la réponse modèle est comparé avec tout mot dans la réponse d'apprenant, en créant une matrice de taille N*M où N est le nombre de mots dans la réponse modèle et M étant le nombre de mots dans la réponse d'apprenant. Nous cherchons à former une matrice où chaque ligne représente un mot de la réponse modèle et chaque colonne représente un mot de la réponse d'apprenant. La valeur d'une case Mat[i j] représente la similarité entre le mot i et le mot j obtenue en appliquant le cosinus aux vecteur des deux mots

Après avoir construit la matrice de similarité entre mots, nous appliquons la formule suivante pour obtenir une similarité entre la réponse modèle et la réponse d'apprenant :

$$Sim(P,R) = \frac{1}{2}\left(\frac{\sum_{w\ \in\{P\}}(maxSim(w,R)*idf(w))}{\sum_{w\ \in\ P}\ idf(w)} + \frac{\sum_{w\ \in\{R\}}(maxSim(w,P)*idf(w))}{\sum_{w\ \in\ R}\ idf(w)}\right)$$

Où maxSim(w,R) représente la similarité max entre le mot w de P et tous les mots de R. La pondération idf est celle déjà présentée dans les sections précédentes.

Nous avons modifié cette approche en remplaçant dans la formule de calcul de la similarité Sim(P,R), la pondération IDF par les nouvelles pondérations TFMinMAX et TF-IDF.

2.3. Calcul des notes et évaluation

Le module « Calcul des notes et évaluation » est le dernier module de du système. Il contient 4 sous modules, 3 pour la génération de score et un 4eme pour l'évaluation des résultats ; une évaluation faite en lot sur l'ensemble du dataset.

A- Le module Kmeans :

Le modules Kmeans est l'un des trois modules que nous avons mis en place pour passer des similarités obtenues vers la note finale. Il est comme son nom l'indique implémenté par l'algorithme de classification non supervisé K-means [39] [40].

Traitant les datasets présentés auparavant, où les notes varient de 1 à 5, nous avons choisi la valeur K=11 pour permettre un pas de 0.5 dans cet intervalle. De ce fait les 11 classes se font affectées une note en partant de 0 pour la classe contenant les similarités les plus base à 5. Ce qui revient à noter sur une échelle de 11 notes (0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5).

B- Le module X5 :

L'implémentation de ce module se résume en une opération mathématique simple : la multiplication. En effet, tous les modules de calcul de similarité revoient une valeur similarité comprise dans l'intervalle 0 et 1, les deux valeurs incluses. Pour obtenir une note sur une échelle de 0 à 5, il suffit de multiplier cette similarité par 5.

C- Le module KNN :

Contrairement aux deux autres modules chargés de la même tâche, celui-là implémente l'algorithme de classification supervisé des K plus proches voisins (K-NN ou K-PPV) [43] qui doit être entrainé sur un jeu de données pour pouvoir par la suite prédire la classe d'une nouvelle entrée. Nous avons utilisé la moitié du dataset de Molher et Mehalcea comme donnée d'apprentissage (d'entrainement).

- Le module d'évaluation :

L'évaluation d'un système implémenté ou d'une approche proposée est indispensable pour estimer le succès d'une recherche [44]. Il devient primordial d'accorder un rôle central aux métriques d'évaluation qui consiste à comparer un résultat produit avec des résultats corrects attendus. L'analyse de plusieurs situations d'évaluation dans notre cas, illustre l'importance d'un choix cohérent des métriques et de l'utilisation conjointe de plusieurs métriques. En essayant

d'analyser les résultats de ce travail, nous avons été confrontés à la détermination de la métrique à utiliser pour évaluer les scores obtenus par rapport aux scores manuels fournis. la décision de choix de métriques a été influencée par le datasets et les travaux connexes qui ont utilisé ce mêmes datasets. La corrélation de Pearson [45] est la métrique la plus fréquemment utilisée par les recherches dans ce domaine. C'est le cas aussi des différents datasets utilisés dans ce travail. Bien qu'elle ne soit pas citée et utilisée dans la majorité des travaux connexes, nous avons choisi d'inclure conjointement au coefficient de Pearson, l'erreur quadratique moyenne (Root Mean Squared Error (RMSE) [15]) pour quantifier la différence (ou le décalage) entre le résultat(score) obtenu par le système et celui obtenu par l'expert humain.

A- Coefficient de Pearson(r) :

En statistiques, étudier la corrélation entre deux ou plusieurs variables statistiques numériques, c'est étudier l'intensité de la liaison ("proportionnalité") qui peut exister entre ces variables. La mesure de la corrélation linéaire entre les deux se fait alors par le calcul du coefficient de corrélation linéaire, noté r. Ce coefficient est égal au rapport de leur covariance et du produit non nul de leurs écarts types. Le coefficient de corrélation est compris entre -1 et 1

Corrélation	Négative	Positive
Faible	de -0,5 à 0,0	de 0,0 à 0,5
Forte	de −1,0 à −0,5	de 0,5 à 1,0

Tableau 4 Signification des valeurs la corrélation de Pearson

Plus le coefficient est proche des valeurs extrêmes -1 et 1, plus la corrélation linéaire entre les variables est forte ; on emploie simplement l'expression « fortement corrélées » pour qualifier les deux variables. Une corrélation égale à 0 signifie que les variables ne sont pas corrélées linéairement. Le coefficient de corrélation est multiplié par 100 pour exprimer un pourcentage de corrélation. Dans notre cas les variables statistiques à considérer sont celles définies dans deux vecteurs l'un contenant les valeurs de scores entre les couples de réponses du dataset (réponse de l'étudiant, réponse modèle de l'enseignant) calculés automatiquement,

le deuxième vecteur contient les scores, pour les mêmes couples de réponses, calculées par l'expert humain. L'objectif dans ce travail revient à maximiser ce coefficient.

B- Erreur quadratique RMSE (Root Mean Squared Error (RMSE)):

L'erreur quadratique moyenne permet de quantifier une mesure synthétique de l'erreur globale commise. Pour calculer l'erreur quadratique moyenne RMSE, les erreurs individuelles sont tout d'abord élevées au carré, puis additionnées les unes aux autres. On divise ensuite le résultat obtenu par le nombre total d'erreurs individuelles, puis on en prend la racine carrée. L'erreur quadratique est probablement le critère quantitatif le plus utilisé pour comparer des valeurs calculées (ici les scores ou notes automatiques) et valeurs observées (scores manuels attribués par l'expert humain. C'est cette fonction que nous tentons de minimiser dans le cadre de ce travail.

En conclusion, l'évaluation de nos approches correspond à trouver la meilleure minimisation de l'erreur quadratique avec une maximisation du coefficient de corrélation.

2.4. Combinaison des approches

Nous avons combiné les similarités générées par plusieurs des modèles de calcul de similarité afin d'améliorer les résultats obtenus. Les critères de combinaison considérés sont les suivants :

- Combinaison locale « Combine ALL-Best » : Nous avons pour chaque approche, et pour chaque modèle WE combiné toutes les méthodes ayant en résultat une corrélation de Pearson supérieure à 50%

- Combinaison avec d'autres mesures sans recours aux WE

Une opportunité s'est offerte à nous pendant la réalisation de ce travail, à savoir, qu'un monôme de notre spécialité travaille aussi dans le même projet que nous sur l'évaluation automatique des réponses courtes appliquées à la langue anglaise en utilisant des mesures de similarité sémantiques n'utilisant pas les Word Embedding.

- Abdellaoui Salah, ayant pour thème : Mesures de similarité sémantique pour un système d'évaluation automatique des réponses courtes : Application à la langue anglaise [46]

Nous avons en premier lieu uniformisé nos sorties par rapport à ses sorties générés de son module de calcul de notes. Nous avons ensuite combiné notre meilleur Combine-Best avec le leur pour estimer l'amélioration apportée par une telle hybridation.
Les résultats de cette hybridation sont présentés dans le chapitre 4.

Conclusion

Au cours de ce chapitre, nous venons de présenter les principes du système. Ce dernier est un ensemble de plusieurs modules intégrés pour effectuer une évaluation automatique de réponses courtes, Ce processus nécessite un prétraitement du corpus dont il sera utilisé pour la génération des pondérations dans le but de tester son impact. Le processus suivant se charge du calcul de similarité qui sera convertis en score par la suite. Deux métriques d'évaluation ainsi qu'un dataset seront utilisés pour déterminer le degré de son efficacité et sa fiabilité. Les résultats des tests ainsi que ceux des hybridations seront présentés dans le chapitre suivant.

Chapitre 3

Synthèse expérimentale et Résultats

Dans ce chapitre, nous montrons les expériences et les résultats des tests réalisés afin d'estimer la qualité du système et sa performance par rapport à l'anglais , ainsi nous discutons les résultats obtenus. Nous présentons dans la suite, une description des étapes menées dans le processus d'expérimentation, une description des ressources matérielles et logicielles utilisées ainsi qu'une description des résultats obtenus.

1. Démarche expérimentale

Après avoir défini les grandes lignes de l'approche méthodologique dans le chapitre 3, nous allons voir à présent plus en détails le processus d'application de cette dernière :

1.1. Prétraitement des textes

Une fois l'acquisition des ressources terminée, nous avons entamé le traitement de ces dernières :

1.1.1. Les datasets

Le Dataset de Molher et Mehalcea est sous format TXT, il se compose de trois tâches de sept questions, chacune étant donnée à un cours d'initiation à l'informatique à l'Université de North Texas. Les données sont au format texte brut. Chaque tâche comprend la question, réponse d'enseignant et un ensemble de réponses d'élèves avec les notes moyennes de deux annotateurs inclus. Les deux annotateurs ont été invités à noter l'exactitude sur un entier échelle de 0 à 5. La corrélation inter annulaire sur l'ensemble de données était de 0,6443. en utilisant le coefficient de Pearson. Un identifiant étudiant unique est également fourni.

Le format des fichiers d'affectation est le suivant :
Pour chaque question de la assignation, les 4 premières lignes sont, respectivement, une ligne sentinelle contenant uniquement des caractères '#', la ligne de question commençant par la chaîne "Question :", la ligne de réponse commençant par la chaîne "Answer:", et une ligne blanche. Les N lignes suivantes (où N est le nombre d'élèves qui ont soumis la mission) contiennent chacun la note moyenne donnée pour la réponse, un identifiant étudiant unique associé à la réponse (entre parenthèses []), et enfin la réponse brute elle-même. Enfin, après toutes les lignes de réponse, un blanc précède le début de la question suivante.

1.1.2. Les corpus :

Ayant fait recours aux méthodes statistiques, nous avons effectué plusieurs manipulations sur le corpus BBC, et Gutenberg, pour garantir des valeurs statistiques fiables, notamment :

- Un nettoyage de contenu : en supprimant de chaque fichier les caractères de ponctuation, et les signes diacritiques s'ils existent.

- Un stemming des documents : en appliquant pour une copie du corpus un stemming léger (avec Porter stemmer) et pour une autre copie un stemming un peu plus lourd (avec Snowball stemmer), nous avons donc, en final, 3 copies du corpus : un corpus brut et deux corpus stemmés.
- Une normalisation des textes

Concernant le choix du corpus, nous avons choisi d'utiliser un corpus de domaine [44] spécifique qui parle de l'informatique générale et surtout des langages des programmation, il prend une taille de 2.14 Mb ainsi qu'un nombre de mots exhaustive égale a 10k répartie en 463 documents.

1.2. Génération des pondérations des mots :

1.2.1. Par un calcul de fréquence :

Les méthodes statistiques permettent d'identifier les mots les plus importants d'un texte, nous avons décrit le principe de ces derniers dans le chapitre 3 ainsi que les pondérations que nous avons utilisées, à savoir IDF, TFlog, TFminmax, TF-IDFminmax.

Comme on à pu le voir en expliquant le fonctionnement du module « calcul statistique » au chapitre 3, la génération des fréquences est basée sur des formules mathématiques qui nécessitent des données numériques, en s'aidant du package « GenSim » de python, nous avons :

- Reconstitué le corpus :
    - Pour chaque document, un vecteur contenant les mots de celui-ci a été créé
    - Chaque vecteur créé est ajouté à un autre vecteur corpus formant une matrice du corpus au complet.
- Créé un dictionnaire : chaque mot unique de la matrice se voit attribuer un identificateur numérique unique. Nous avons fait appel à la méthode « Dictionary ». A partir de là, nous travaillons avec les identificateurs des mots au lieu des mots eux-mêmes.
- Calculé les IDFs : les IDFs ont été récupérés depuis l'instance Dictionary créé à l'étape précédent par la méthode « dfs » de cette dernière.
- Calculé le nombre d'apparition de chaque élément : une représentation « sac à mot » où « bag of words » du corpus a été effectué grâce à la méthode « doc2bow » afin de faciliter les calculs.

Une fois ces calculs terminés, nous avons pu implémenter les formules mathématiques, les appliquer pour notre corpus numérisé, et enfin, exporter les résultats avec les formats décris dans la description du l'outil calcul de fréquence.
Concernant le choix du corpus, malgré la grande variation des corpus en langue anglaise, nous nous somme contenté de deux corpus vu l'incapacité de nos appareils a gérer des corpus trop volumineux, dont un corpus assez grand ' Gutenberg ' dans la limite des performances du matériel, et un autre plus petit ' BBC ' nous avons commencé par évaluer les corpus en notre disposition, le processus d'évaluation étant comme suit :

- Chaque corpus à été nettoyé, stemmé puis normalisé.
- Par la suite nous avons généré les fréquences des mots de chaque corpus
- Puis nous avons testé ces fréquences avec la méthode SomVec (décrite plus loin) sur nos datasets.

Les résultats obtenus indiquant clairement ' Gutenberg ' comme meilleur corpus vu son contenu riche en mots, nous l'avons donc retenu pour la suite de notre expérimentation.

1.2.2. Par un analyseur morphosyntaxique :

Contrairement aux approches statistiques où chaque mot est représenté par une seule valeur qui détermine son importance par rapport au corpus, la pondération orienté partie du discoure (POS) détermine le poids d'un mot par rapport à sa fonction dans la phrase, ce qui fait que la valeur d'un mot peut varier d'une occurrence à une autre du terme en question. Pour cette raison, nous avons adopté une représentation en tuple pour joindre chaque mot avec la partie du discours qu'il assure dans la phrase. Par exemple : (compilate, verb)
Nous avons utilisé cette représentation pour créer de nouvelles versions de nos datasets où chaque phrase est représentée par un tableau dont chaque élément est un tuple (mot, POS du mot). Nous avons généré ces nouveaux datasets en partant des datasets stemmés et non normalisé, Voici une illustration du traitement :
«constructor initializes data » → [(constructor, Noun) , (initializes, Verb) , (data , Noun)
L'identification des POS a été effectuée grâce à l'analyseur morphosyntaxique « NLTK POS »[47] qui compte les tags suivants :

CC	Coordinating conjunction	CD	Cardinal number	MD	Modal
FW	Foreign word	DT	Determiner	NNP	Proper noun, singular
IN	Preposition	JJR	Adjective comparative	POS	Possessive ending
JJS	Adjective, superlative	JJ	Adjective	RB	Adverb
NN	Noun, singular or mass	LS	List item marker	RP	Particle
NNPS	Proper noun, plural	NNS	Noun, plural	WP$	Possessive
PRP	Personal pronoun	PRP$	Possessive pronoun	WP	Wh-pronoun
RBR	Adverb, comparative	RBS	Adverb, superlative	WDT	Wh-determiner
UH	Interjection	VB	Verb, base form	WRB	Wh-adverb
VBG	Verb, gerund or present participle	VBN	Verb, past participle	VBZ	Verb, 3rd person singular present
VBP	Verb, non3rd person singular present	VBD	Verb, past tense	PDT	Predeterminer

Tableau 5 Liste des tags

Nous avons simplifié les classes de sorties comme suite :

- VB VBP VBD VBG VBN VBZ → Considérées comme un verbe et représentées par un V
- NN NNS NNP NNPS → Considérées comme un nom et représentées par un N
- JJ JJR JJS RB RBR RBS → Considérées comme un adjectif et représentées par un AJ

Appliqué à notre exemple :

[(constructor, Noun) , (initializes, Verb) , (data , Noun)] → [(constructor, N) , (initializes, V) , (data , N)]

Une fois que ce processus terminé, nous générons nos nouvelles versions des datasets avec cette représentation tabulaire en les exportant sous format. npy grâce aux package « numpy ». Au besoin d'une pondération POS, nous faisons appel à ces fichiers. npy pour :

- Récupérer les mots
- Chercher le vecteur de chaque mot dans nos modèles des WE après l'avoir normalisé selon le modèle en question.
- Multiplier le vecteur du mot par 0.2 si POS='V', 0.35 si POS='N' sinon si POS = 'AJ' par 0.45

1.3 Calcul de similarité :

Etant le noyau du système, nous avons utilisé plusieurs méthodes de calcul de similarité représenté par des modules : SOMVEC, Calcul matriciel, Mehalcea modifié.

- SOMVEC : Ce module regroupe toutes nos méthodes dérivées de la somme des vecteurs qui sont :
    - SomVec : Somme des vecteurs unitaire où $\beta = 1$
    - SomVecIDF, SomVecTFMINMAX, SomVecTFLOG, SomVecTFIDFminmax, SomVecPOS, SomVecMIXTE où β varie selon les mesures générées par les méthodes dont le nom est porté par la méthode.
- Calcul matriciel : Ce module comporte deux méthodes, l'une dérivé de l'autre :
    - MatSim : Calcul matriciel mot à mot où $\beta = 1$
    - MatSimP : Calcul matriciel mot à mot où β varie selon les mesures générées par la combinaison des méthodes TFMINMAX et POS qu'on notera pondération Mixte.
- Mehalcea modifiée : Ce module regroupe toutes nos méthodes dérivées de l'approche de Mahalcea
    - Mahalcea où β varie selon IDF, qui est la formule de base.
    - MahalceaIDF, MahalceaTFMINMAX, MahalceaTFLOG, MahalceaTFIDFMINMAX, MahalceaPM où β varie selon les mesures générées par les méthodes dont le nom est porté par la méthode.

L'implémentation des modules a été faite en python, deux points importants :

- La récupération des vecteurs des mots :

Toutes nos méthodes utilisent les modèles Word Embeddings. Pour faciliter cette manipulation nous avons utilisé plusieurs méthodes de gensim :

- KeyedVectors.load_word2vec_format : pour charger les modèles de Google News en mémoire.
- gensim.models.Word2Vec.load : pour charger les modèles des WE FastText en mémoire.
- Une fois le modèle chargé, nous pouvons récupérer le vecteur d'un mot directement par l'instruction : vecteur = modele[mot].
- model.similarity(x,y): qui calcul la similarité cosinus entre deux mots passés en paramètres.

- La récupération des pondérations : Nécessite de passer en paramètre de la méthode le dictionnaire du vocabulaire et le dictionnaire de pondération, par exemple IDF.p générés après le calcul des fréquences. Le dictionnaire étant de la forme { ID : mot } a besoin d'être inversé, une fois passé à la forme { Mot : ID } nous pouvons chercher l'ID d'un mot dans le dictionnaire puis la pondération de ce mot grâce a son ID dans le dictionnaire de pondération, qui est quant à lui, de la forme { ID : Valeur }

Une fois tous nos modèles de calcul de similarité implémentés, nous avons réalisé une série de tests sur chaque modèle. Considérons par exemple le modèle SomVec : nous avons évalué notre dataset par ce module, une fois avec les copies non stemmés, une 2eme fois avec les copies stemmés avec Porter et une 3eme fois avec les copies stemmés avec Snowball. Les résultats sont, après conversion en notes, rapportés dans la partie suivante.

1.4 Calcul des notes et évaluation :

Les modèles de calcul de similarité renvoient des valeurs entre 0 et 1 les deux valeurs inclues. Ces valeurs doivent être converti en note afin d'évaluer correctement les modèles. Nous avons fait recours à deux algorithmes de classification :

- cKmeans (k-means Clustering in One Dimension) : une implémentation du Kmeans classique pour des données d'une seul dimension.
- KNN : nous avons utilisé le KNN du package « Sklearn » qu'on a entrainé en divisant notre dataset sur deux ensembles, un jeu d'entrainement de 435 couples de réponses et un jeu de test des 174 couples restants, en calculant la similarité entre les couples de phrases du dataset puis nous avons attaché à chaque similarité calculée la note manuelle accordée au couple de phrase d'où elle a été calculée. Les données générées ont été introduites comme bases d'apprentissage pour ce module
- Hormis les algorithmes de classification, nous avons implémenté une fonction « X5 » basique réalisant une multiplication de la similarité par 5 pour avoir une note.

Nous avons testé les 174 couples aussi avec le kmeans et X5 pour avoir un test significatif et cohérent, une fois nos modules de notation prêts, nous avons converti toutes les similarités en notes avec les deux modules Kmeans et X5, le KNN quant à lui, a été rapidement ignoré aux vues de ces résultats exposé dans la partie suivante.

2. Ressources matérielles et logicielles

Les scripts ont été développées en utilisant Python 3.6.
Nous avons rencontré une contrainte matérielle majeure puisque nos machines (4 GO RAM) ne pouvaient trainer des fichiers de WE de l'ordre de 6GO. Notre problème a été réglé une fois que nous l'avons augmenté jusqu'à 12 GO de RAM, les caractéristiques suivantes (de nos PC) peuvent être considérées comme configuration minimale pour le bon fonctionnement de nos scripts :

Processeur : i5-3210M / 2.50 GHz
RAM: 12GO
OS: Windows 10

- **Outil de calcule**

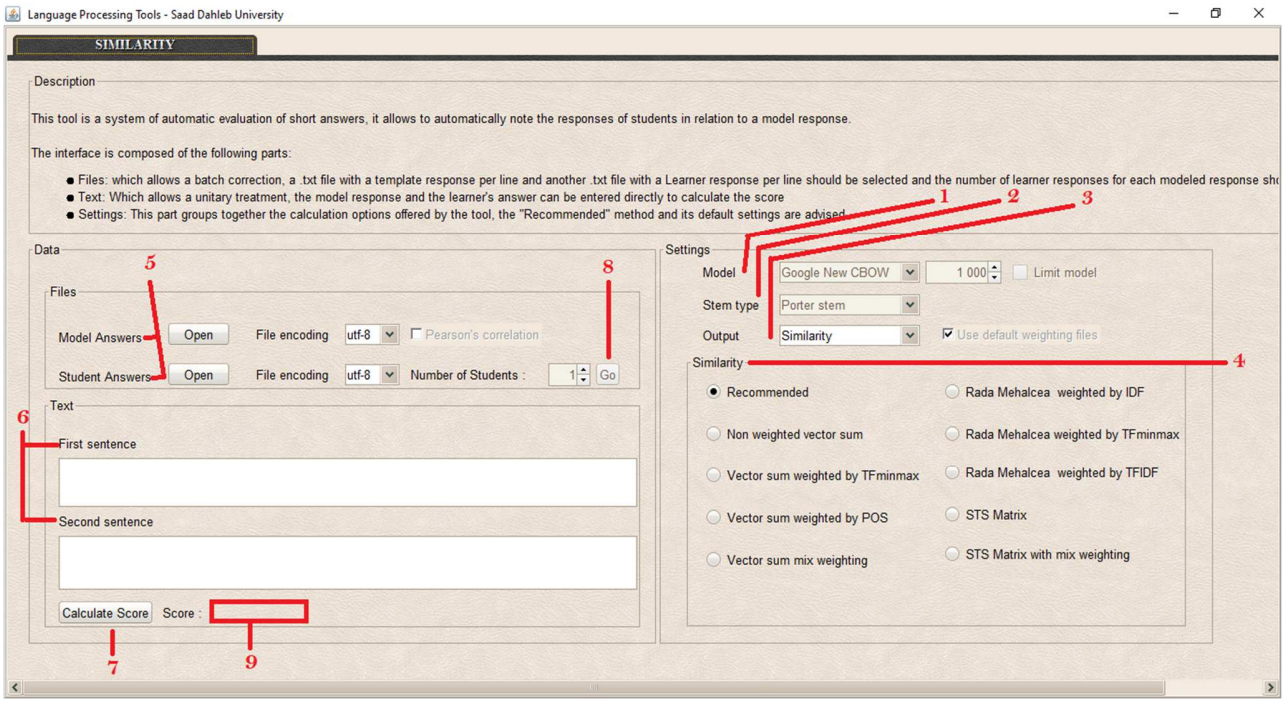

Figure 12 outil de calcule

1) Sélection du model du word embedding (Google News CBOW, FastText Skip-gram, FastText CBOW).
2) Sélection du Stemmer (Sans stem, Porter stem, Snowball stem).
3) Sélection du type de sortie (Similarité, Score, Coefficient de Pearson).
4) Sélection de l'approche de calcule de similarité.
5) Sélection du ficher contenant les réponses modèle ainsi que celui contenant les réponses des étudiants.
6) Zones de saisie des deux expressions pour un calcule de similarité pour ces deux dernières seulement.
7) Exécution du calcule de similarité seulement pour les deux expressions dans (6).
8) Execution du calcule de similarité pour les deux fichiers sélectionnés déjà dans (5) et ceci après la sélection du nombre de réponses d'étudiants à comparer pour chaque réponse modèle.
9) Zone d'affichage du résultat.

3. Synthèse expérimentale

- **Résultats**

L'analyse expérimentale quantitative a été menée pour chaque modèle de calcul de similarité dans chacune des 3 approches. Pour chaque modèle nous avons généré autant de « RUN » que de couples de réponses à évaluer.
Les notes obtenues automatiquement sont confrontées aux notes attribuées manuellement afin de calculer la corrélation de Pearson et l'erreur quadratique moyenne.

Légende :
CP : Coefficient de Pearson
EQ : Erreur quadratique
MO : Matrice d'ordre

- **L'impact des Stemmers**

Nous avons appliqué les Stemmers sur toutes les méthodes, et nous constatons que les résultats qui ont été généré a partir d'un dataset non stemmé sont nettement supérieurs que les autres stemmé, on a donc affiché les trois tableaux ci-dessous comme exemple de l'impact des stemmer sur la méthode SOMVEC :

Stemmer	Modèle WE	SOMVEC	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 45,37 EQ: 1,77	CP: 44,92 EQ: 1,55
	FastText (Skip-Gram)	CP: 42,39 EQ: 1,72	CP: 41,67 EQ: 1,16
	FastText (CBOW)	CP: 48,20 EQ: 1,84	CP: 46,82 EQ: 1,30

Tableau 6 Le modèle de similarité : SOMVEC sans Stem

Nombre de mots manquants :

Google (CBOW) : 53

FastText (Skip-Gram) : 0

FastText (CBOW) : 0

Stemmer	Modèle WE	SOMVEC	
		Kmean	X5
Porter Stem	Google (CBOW)	CP: 41,85 EQ: 1,99	CP: 41,19 EQ: 1,80
	FastText (Skip-Gram)	CP: 42,74 EQ: 1,91	CP: 41,43 EQ: 1,22
	FastText (CBOW)	CP: 46,50 EQ: 1,82	CP: 46,02 EQ: 1,41

Tableau 7 Le modèle de similarité : SOMVEC avec Porter Stem

Nombre de mots manquants :

Google (CBOW) : 228

FastText (Skip-Gram) : 0

FastText (CBOW) : 0

Stemmer	Modèle WE	SOMVEC	
		Kmean	X5
Snowball Stem	Google (CBOW)	CP: 41,07 EQ: 2,03	CP: 41,36 EQ: 1,80
	FastText (Skip-Gram)	CP: 43,17 EQ: 1,90	CP: 41,60 EQ: 1,22
	FastText (CBOW)	CP: 46,67 EQ: 1,81	CP: 46,44 EQ: 1,40

Tableau 8 Le modèle de similarité : SOMVEC avec Snowball Stem

Nombre de mots manquants :

Google (CBOW) : 213

FastText (Skip-Gram) : 0

FastText (CBOW) : 0

Nous constatons aussi que seul les résultats du WE de FastText (Skip-Gram) s'améliore en utilisant les stemmers dans cet exemple , or, le nombre de mots manquant augmente en utilisant les stemmers, mais aussi la majorité des tests ont donnés de bons résultats en étants non stemmés, donc on affichera dans les tableaux qui suivent, les meilleurs résultats, en l'occurrence ceux non stemmé.

- **Impact des modules d'évaluation**

Nous avons appliqué les tests sur 174 couples de réponses comme nous l'avant décrit précédemment et ce pour les trois modules d'évaluations pour pouvoir préserver les mêmes conditions de tests :

Approche	Modèle	Datasets	SomVec non pond		
			Kmeans	X5	KNN
sans stemm	google(CBOW)	Mohler et Mihalcea	CP: 52.61 EQ: 1.88	CP: 51.38 EQ: 1.61	CP: 30.00 EQ: 1.45

	fasttext(SKIPGRAM)		CP: 49.68 EQ: 1.90	CP: 45.86 EQ: 1.14	CP: 28.05 EQ: 1.62
	fasttext(CBOW)		CP: 60.49 EQ: 1.71	CP: 60.71 EQ: 1.18	CP: 32.66 EQ: 1.42

Tableau 9 Le modèle de similarité : SOMVEC

Approche	Modèle	Datasets	SomVec non pond		
			Kmeans	X5	KNN
porter stem	google(CBOW)	Mohler et Mihalcea	CP: 50.30 EQ: 1.80	CP: 51.38 EQ: 1.61	CP: 29.45 EQ: 1.80
	fasttext(SKIPGRAM)		CP: 48.20 EQ: 1.45	CP: 44.75 EQ: 1.53	CP: 27.52 EQ: 1.70
	fasttext(CBOW)		CP: 59.92 EQ: 1.19	CP: 60.15 EQ: 1.40	CP: 31.40 EQ: 1.38

Tableau 10 Le modèle de similarité : SOMVEC avec Porter Stem

Approche	Modèle	Datasets	SomVec non pond		
			Kmeans	X5	KNN
snowball stem	google(CBOW)	Mohler et Mihalcea	CP: 50.02 EQ: 1.40	CP: 50.80 EQ: 1.45	CP: 28.98 EQ: 1.60
	fasttext(SKIPGRAM)		CP: 48.21 EQ: 1.50	CP: 43.93 EQ: 1.16	CP: 27.21 EQ: 1.49
	fasttext(CBOW)		CP: 59.41 EQ: 1.49	CP: 59.70 EQ: 1.26	CP: 31.22 EQ: 1.59

Tableau 11 Le modèle de similarité : SOMVEC avec Snowball Stem

- *Nous constatons que le module de calcule de score KNN a donné de mauvais résultats, ceci peut être du au fait que cette approche supervisée a besoin de plus de ressources d'entrainement et de jeux de test*

- **Les résultats des méthodes pondéré et POS**

Stemmer	Modèle WE	SOMVEC-POS	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 47,18 EQ: 1,71	CP: 48,08 EQ: 1,58
	FastText (Skip-Gram)	CP: 44,55 EQ: 1,63	CP: 47,86 EQ: 1,16
	FastText (CBOW)	CP: 50,91 EQ: 1,59	CP: 51,80 EQ: 1,31

Tableau 12 Le modèle de similarité : SOMVEC-POS

- *Nous constatons une nette amélioration au niveau du CP en appliquant POS*

Stemmer	Modèle WE	SOMVEC-IDF	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 41,51 EQ: 1,96	CP: 42,04 EQ: 1,71
	FastText (Skip-Gram)	CP: 39,87 EQ: 1,93	CP: 38,94 EQ: 1,24
	FastText (CBOW)	CP: 46,72 EQ: 1,89	CP: 46,85 EQ: 1,37

Tableau 13 Le modèle de similarité : SOMVEC-IDF

- *Nous constatons une baisse flagrante au niveau du CP et une hausse au niveau de l'EQ en appliquant la pondération IDF*

Stemmer	Modèle WE	SOMVEC-TFminmax	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 44,46 EQ: 1,97	CP: 45,00 EQ: 1,63
	FastText (Skip-Gram)	CP: 42,13 EQ: 1,78	CP: 40,57 EQ: 1,18

	FastText (CBOW)	CP: 48,30 EQ: 1,85	CP: 47,63 EQ: 1,33

Tableau 14 Le modèle de similarité : SOMVEC-TFminmax

- *CP plus bas par rapport a SOMVEC et EQ plus hausse*

Stemmer	Modèle WE	SOMVEC-Tflog	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 44,71 EQ: 1,97	CP: 45,67 EQ: 1,62
	FastText (Skip-Gram)	CP: 42,58 EQ: 1,78	CP: 40,82 EQ: 1,18
	FastText (CBOW)	CP: 49,29 EQ: 1,82	CP: 48,56 EQ: 1,31

Tableau 15 Le modèle de similarité : SOMVEC-TFlog

- *La pondération Tflog ne donne également pas de bons résultats vu le CP toujours bas*

Stemmer	Modèle WE	SOMVEC-Tfidf	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 41,35 EQ: 1,96	CP: 40,56 EQ: 1,83
	FastText (Skip-Gram)	CP: 39,56 EQ: 1,88	CP: 39,09 EQ: 1,38
	FastText (CBOW)	CP: 46,79 EQ: 1,98	CP: 47,38 EQ: 1,43

Tableau 16 Le modèle de similarité : SOMVEC-TFidf

- *La pondération TFidf donne de mauvais résultats puisqu'il s'agit du produit de deux mauvaises pondérations TF*IDF. Le résultat est prévisible.*

Stemmer	Modèle WE	Mihalcea-IDF	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 47,97 EQ: 2,40	CP: 47,97 EQ: 1,99
	FastText (Skip-Gram)	CP: 48,12 EQ: 2,13	CP: 46,56 EQ: 1,62

	FastText (CBOW)	CP: 49,50 EQ: 2,34	CP: 49,15 EQ: 1,74

Tableau 17 Le modèle de similarité : Mihalcea-IDF

- *Nous constatons une amélioration par rapport a la méthode SOMVEC au niveau du CP, or, l'EQ a augmenté considérablement*

Stemmer	Modèle WE	Mihalcea-Tfminmax	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 49,94 EQ: 2,06	CP: 48,39 EQ: 2,00
	FastText (Skip-Gram)	CP: 49,27 EQ: 2,09	CP: 48,13 EQ: 1,63
	FastText (CBOW)	CP: 50,02 EQ: 2,16	CP: 50,93 EQ: 1,74

Tableau 18 Le modèle de similarité : Mihalcea-TFminmax

- *Nous constatons encore une amélioration du CP mais aussi une baisse de l'EQ*

Stemmer	Modèle WE	Mihalcea-Tflog	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 50,08 EQ: 2,21	CP: 49,76 EQ: 1,96
	FastText (Skip-Gram)	CP: 49,70 EQ: 2,09	CP: 48,41 EQ: 1,61
	FastText (CBOW)	CP: 50,74 EQ: 2,15	CP: 51,80 EQ: 1,71

Tableau 19 Le modèle de similarité : Mihalcea-TFlog

- *Le CP a augmenté et touche désormais les 50 pour le WE de Google (CBOW) par contre l'EQ a un peu augmenté par rapport à la pondération TFminmax*

Stemmer	Modèle WE	Mihalcea-Tfidf	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 47,84 EQ: 2,26	CP: 47,22 EQ: 2,06
	FastText (Skip-Gram)	CP: 47,14 EQ: 2,15	CP: 46,43 EQ: 1,66
	FastText (CBOW)	CP: 48,55 EQ: 2,29	CP: 48,89 EQ: 1,79

Tableau 20 Le modèle de similarité : Mihalcea-TFidf

- *Comme pour la méthode SOMVEC-Tfidf, les résultats sont plus bas, toujours en raison du produit de deux mauvaises pondérations TF*IDF*

- **Pondération Mixte**

En ce qui concerne la pondération mixte, on a combiné les pondérations qui ont donné les meilleurs résultats, en l'occurrence la pondération TFlog, avec POS, cette approche a augmenté les résultats mais pas de façon considérable, on donnera un exemple de comparaison entre les résultats de la matrice d'ordre MO et ceux de la MO-Mixte, et on affichera ensuite le reste des pondération Mixte :

Stemmer	Modèle WE	MO	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 49,48 EQ: 2,36	CP: 46,06 EQ: 2,27
	FastText (Skip-Gram)	CP: 49,05 EQ: 2,32	CP: 46,07 EQ: 1,99
	FastText (CBOW)	CP: 49,36 EQ: 2,36	CP: 48,22 EQ: 2,09

Tableau 21 Le modèle de similarité : Matrice d'ordre

- *Nous constatons à première vue que les résultats sont meilleurs que ceux de SOMVEC, donc on suppose que pour la pondération Mixte sa sera meilleurs, or l'EQ reste assez haute*

Stemmer	Modèle WE	MO-Mixte	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 51,48 EQ: 2,35	CP: 50,45 EQ: 2,31
	FastText (Skip-Gram)	CP: 51,72 EQ: 2,03	CP: 51,27 EQ: 2,01
	FastText (CBOW)	CP: **52,09** EQ: 2,06	CP: 51,60 EQ: 2,11

Tableau 22 Le modèle de similarité : Matrice d'ordre Mixte

- *Jusqu'à maintenant on note les meilleurs résultats par rapport aux précèdent on verra par la suite si les hybridations vont encore les améliorer*

Stemmer	Modèle WE	SOMVEC-Mixte	
		Kmean	X5
Sans Stem	Google (CBOW) 53	CP: 46,83 EQ: 1,83	CP: 47,93 EQ: 1,63
	FastText (Skip-Gram) 0	CP: 44,73 EQ: 1,60	CP: 47,49 EQ: 1,18
	FastText (CBOW) 0	CP: 51,07 EQ: 1,61	**CP: 51,75 EQ: 1,34**

Tableau 23 Le modèle de similarité : SOMVEC-Mixte

Stemmer	Modèle WE	Mihalcea-Mixte	
		Kmean	X5
Sans Stem	Google (CBOW)	CP: 49,99 EQ: 2,16	CP: 49,41 EQ: 1,94
	FastText (Skip-Gram)	CP: 49,98 EQ: 2,01	CP: 49,26 EQ: 1,58
	FastText (CBOW)	CP: 49,58 EQ: 2,19	CP: 48,51 EQ: 1,71

Tableau 24 Le modèle de similarité : Mihalcea-Mixte

- *Nous constatons une amélioration pour l'approche mixte pour les deux méthodes, néanmoins la méthode de MO-Mixte reste encore celle qui a donné les meilleurs résultats*

- **Hybridation de nos approches**

Nous avons combiné les similarités générées par plusieurs de nos modèles de calcul de similarité afin d'améliorer les résultats obtenus :

**** Combine ALL-BEST :**

Nous n'avons considéré dans la combinaison que les WE de FastText (CBOW), combinant toutes les méthodes ayant un CP supérieure à 50%

Combine ALL-BEST			
Approche	Modèle	Méthodes utilisées	Résultats
Sans Stem	FastText (CBOW)	-SOMVEC-Mixte -SOMVEC-POS -Mihalcea-TFminmax -Mihalcea-Tflog -MO-Mixte	CP : 53,23 EQ : 1,90

Tableau 25 Combinaison ALL-BEST

- *En combinant les méthodes qui ont données les meilleurs résultats c'est-à-dire ceux dépassant les 50% pour les Coefficient de Pearson et en utilisant bien évidemment le WE de FastText modèle CBOW car ce dernier a donné de bons résultats par rapport aux deux autres, Nous constatons une nette amélioration si ce n'est la plus haute pour l'instant au niveau du CP, puisque les approches testées individuellement ont donné les meilleurs résultats généralement avec ce modèle*

- **Hybridation avec d'autre mesures sans recours aux WE**

Pour améliorer encore les résultats, nous avons combiné notre Combine ALL-BEST avec le meilleur résultat d'un autre monôme travaillant sur la même thématique [44] et une deuxième fois avec deux autres approches syntaxiques qui sont LSC et DICE. En premier lieu uniformisé nos sorties par rapport à ses sorties générés de ses modules de calcul de notes respectifs. Nous

avons ensuite combiné notre meilleur Combine ALL-BEST avec le sien pour estimer l'amélioration apportée par une telle hybridation. Puis nous avons utilisé les méthodes de DICE et LCS sur notre Dataset pour pouvoir faire une hybridation de notre meilleur résultat avec ces approches syntaxiques.

Dans le tableau nous résumons les meilleurs résultats obtenus de cette combinaison :

Combinaison	Résultats
- **Combine ALL-BEST (0.85)** - **Espace sémantique (0.15)**	**CP : 53,81 EQ : 1,83**

Tableau 26 Combinaison avec l'espace sémantique

Combinaison	**Résultats**
- Combine ALL-BEST - DICE	**CP : 53.53 EQ : 1.84**
- Combine ALL-BEST - LCS	**CP : 53.52 EQ :1,83**

Tableau 27 Combinaison avec l'approche syntaxique

- *Pour la combinaison avec l'approche syntaxique nous constatons que les résultats sont un peu plus hausse ce qui confirme une autre fois que l'hybridation permet d'atteindre des résultats nettement meilleurs.*

4. Discussion Globale

Les données du Dataset de Mehalcea sont notées par deux juges humains, de sorte que l'erreur est rapportée par rapport à la moyenne des deux scores. Il a deux L'IAA (Inter Annotator Agreement : Accord inter-annotateurs) parce qu'il évalue question par question (per-question) et il évalue par note globale de l'étudiant par rapport a une évaluation (per-assignement), il est

calculé entre les deux scores fournis par les juges. La corrélation signalée est la corrélation de Pearson

- IAA per-question : consiste a calculer le coefficient de corrélation pour tous les couples de réponses utilisées chaque réponse a part. la valeur est égale a r = 0.6443
- IAA per-assignement : consiste a considérer pour chaque étudiant toute ses réponses dans les trois assignement, ici on mesure la corrélation par rapport a la note globale obtenue par chaque étudiant, et la le IAA est plus grand, il est égale a r = 0.7228

Pour les approches de Mohler les meilleurs résultats sont :

- 0.5099 dans l'évaluation per-question

L'objectif étant donc de se rapprocher et dans les meilleurs des cas dépasser cette valeur.
Notre meilleur résultat a été une hybridation, qui est la combinaison de notre méthode qu'on a appelé Combine ALL-BEST qui consiste à combiner nos meilleurs résultats c'est-à-dire ceux ayant dépassé les 50% au niveau du coefficient de corrélation (Tableau 25) avec les meilleurs résultats obtenus par notre collègue qui a utilisé un espace sémantique (Tableau 26). Le résultat de cette hybridation a dépassé le 0.5099

	Erreur Quadratique Moyenne	Corrélation de Pearson
Approche de Molher	/	0.5099
Combine ALL-BEST/ESP SEM	1.83	0.5381

Tableau 28 Comparaison entre le résultat de Molher et Combine ALL-BEST

- Pour pouvoir encore évaluer les performances du système, nous avons calculé un taux de correspondance avec la correction de l'humain, nous affichons les statistiques de la combinaison All-Best et Espace sémantique :

- Le nombre de fois ou le système a donné la même note que l'humain : 72
  Pourcentage= 11.822660098522167%
- Le nombre de fois ou le système a donné une note ou la différence avec celle de l'humain est de 0<n<=1 : 221
  Pourcentage= 36.28899835796388%

- Le nombre de fois ou le système a donné une note ou la différence avec celle de l'humain est de 1<n<=2 : 174

  Pourcentage= 28.571428571428573%
- Le nombre de fois ou le système a donné une note ou la différence avec celle de l'humain est de 2<n<=3 : 91

  Pourcentage= 14.942528735632184%
- Le nombre de fois ou le système a donné une note ou la différence avec celle de l'humain est de n>3 : 51

  Pourcentage= 8.374384236453203%
- Corrélation de Pearson : 53.64984247977071
- Erreur quadratique moyenne : 1.857174434492184

- Nous avons également mis quelques notes manuelles et automatiques de la meilleure méthode dans un tableau pour pouvoir comparer entre les deux :

Manuelle	**Auto**				
5	2	4,5	1,5	5	4
0	0,5	3,5	1	5	1,5
5	1,5	3	2	3	3,5
5	2,5	5	3,5	2,5	3
5	2,5	4,5	2,5	4,5	4
2	1	4,5	1,5	4,5	4
4	2	5	4,5	5	4
5	4	2,5	0,5	4,5	3,5
5	2,5	5	4,5	3,5	2,5
2	2,5	5	5	4,5	3,5
3	1,5	5	5	4	4
5	5	3,5	2,5	4	4
5	3,5	4,5	3,5	4,5	4
5	5	2	0,5	5	4,5
3,5	4	5	5	4,5	2,5
5	4,5	4,5	4	2	4

Tableau 29 Quelques notes manuelles et leurs équivalents automatiques

- Et voici un graphe représentant ces notes pour ainsi mieux voir la tendance entre le manuel et l'auto :

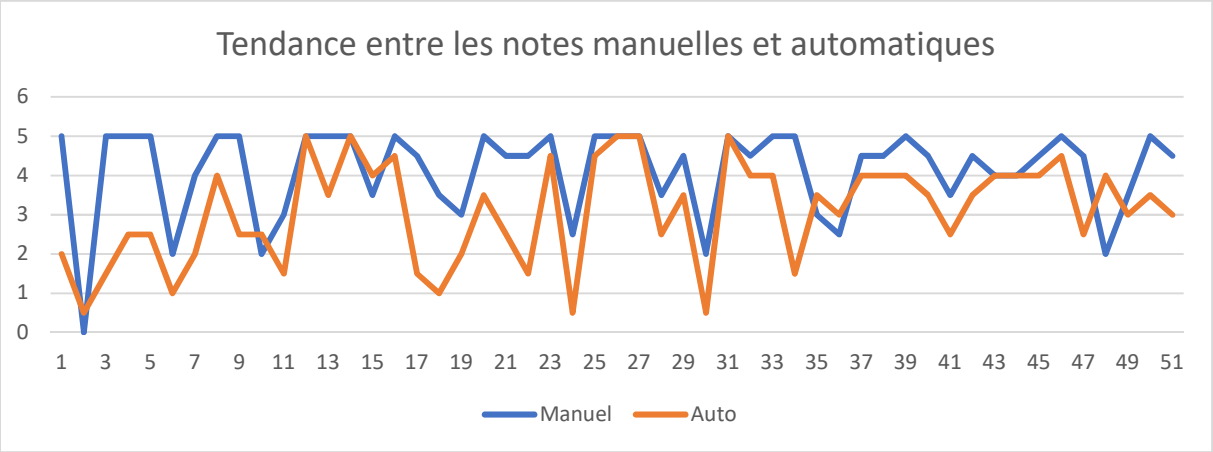

Figure 13 Tendance entre des notes manuelles et automatiques

Au terme de cette synthèse expérimentale, nous aboutissons à plusieurs constatations que nous allons aborder selon leurs aspects suivants :

a) Impact de l'approche de Stemming :

En considérant les résultats obtenus sans considérer les Stem, nous pouvons dire que l'impact de l'approche du stem n'a pas vraiment amélioré les résultats. Ceci est du au fait que l'anglais est une langue qu'on peut qualifier de moins complexe par rapport a d'autre langues, donc le stem peut parfois faire en sorte que le WE ne reconnait pas le mot après avoir été stemmé même s'il le contient.

b) CBOW ou SKIPGRAM ?

Dans notre cas, le CBOW a donné de meilleurs résultats par rapport au Skip-Gram, si on peut faire une description assez simpliste et plutôt naïve de la différence : Comme nous le savons, CBOW apprend à prédire le mot en fonction du contexte. Ou maximisez la probabilité du mot cible en regardant le contexte. Et cela arrive à être un problème pour les mots rares. Par exemple, étant donné le contexte yesterday was a really [...] day le modele CBOW vous dira probablement que le mot est beautiful ou nice. Des mots comme delightful auront beaucoup moins d'attention dans le modèle, car il est conçu pour prédire le mot le plus probable. Ce mot sera lissé sur beaucoup d'exemples avec des mots plus fréquents. D'autre part, le modèle skip-gram est conçu pour prédire le contexte. Étant donné le mot delightful il doit le comprendre et nous dire qu'il existe une probabilité énorme que le contexte soit yesterday was a really [...] day ou un autre contexte pertinent. Avec skip-gram, le mot deslightful n'essaiera pas de rivaliser avec le mot beautiful mais les paires delightful+context seront traitées comme de nouvelles observations

c) Qualité des Word Embedding :

Les Word Embeddings sont très susceptibles aux corpus d'apprentissage. Plus ces derniers sont volumineux, plus les modèles générés sont de meilleure qualité et représentatifs de la langue. C'est d'ailleurs ce qui explique que les WE de FastText sont nettement plus performants que ceux de Google

d) Impact du domaine

Le choix d'un corpus de domaine est dû au fait que le contenu de ce corpus correspond au contenu du dataset, ce qui a résulté que toutes les pondérations des mots du dataset ont été calculé efficacement et ont donné ainsi un meilleur impact de pondération et parallèlement de meilleurs résultats par rapport a l'utilisation d'un corpus normal.

e) Les pondérations :

Une des constatations sûres tirée de cette synthèse est que la pondération, par une approche statistique ou linguistique, est un bon moyen pour améliorer les résultats. La qualité d'une pondération statistique est influencée par la nature du corpus d'apprentissage ; plus les thématiques du corpus sont proches du sujet traité, plus la pondération est fiable, ce qui explique bien sur notre choix d'un corpus de domaine.
La pondération TFlog capture efficacement l'importance des mots par rapport à la langue, la pondération POS est situationnelle, elle ne peut être utilisée avec tous les modules de calcul de similarité, néanmoins, elle a nettement amélioré les résultats

f) Génération de notes (de la similarité au score) :

Nous constatons clairement que les approches non supervisés (Kmeans ici) sont nettement plus performantes que les approches supervisés (KNN pris en exemple). Ceci peut s'expliquer par la qualité des données d'entrainement et, comme par exemple pour le KNN, par le fait qu'il n'est pas adapté aux problèmes dis mal posés comme celui de générer des notes.

g) L'approche hybride :

Nous sommes arrivés à la conclusion que l'hybridation permet d'atteindre des résultats nettement meilleurs. Dans notre cas nous avons combiné manuellement les poids affectés à chaque approche. Toutefois il serait intéressant dans l'avenir d'appliquer un algorithme de régression linéaire pour trouver la meilleure pondération possible des poids des approches à hybrider.

Une autre constatation est la flexibilité des modules de calcul des similarité implémentés avec les WE à être combinés avec d'autre modules issues d'autres approches tels que syntaxique ou sémantique avec une meilleure compatibilité avec les méthodes syntaxique.

Conclusion et Perspectives

L'évaluation automatique est un sujet d'actualité qui n'est pas vraiment facile mais il a beaucoup de bénéfice pour l'avancement de la technologie. Pour cela nous avons axé notre recherche sur l'exploitation des Word Embeddings pour tester l'impact de ces derniers sur la langue anglaise

Notre thème est très vaste et lié avec de nombreux domaines, en conséquence nous avons présenté dans notre mémoire, un état de l'art sur les systèmes d'évaluation, le traitement de la langue anglaise, et les approches de similarité dans le contexte de l'évaluation automatique

Par la suite, nous avons proposé notre approche qui est une continuité d'une approche déjà faite par rapport à la langue arabe, en résultats un système d'évaluation automatique a été adapté à l'anglais et qui se compose de plusieurs modules. Le système est combiné avec d'autres approches syntaxiques et espace sémantique

Nous avons évalué notre système avec un seul dateset, et nous avons obtenue des résultats meilleurs pour le per question par rapport à un système des travaux connexes sur le dataset de Mihalcea, de ce fait, nous avons pu généraliser correctement l'approche. Sachant que les stemmers et les corpus sont des ressources dépendantes de la langue utilisées dans l'approche, cette dernière peut être appliquée sur diverses langues existantes. Ceci exige, bien sûr, une adaptation selon la langue en question : Disponibilité des ressources et outils (corpus et stemmer) ainsi que le prétraitement du corpus approprié à cette langue.

La langue anglaise est très riche en ressources, si ce n'est la plus riche, et même si nos résultats ont dépassé l'objective, ils restent loin d'être optimale dû au fait que pour pouvoir l'exploiter au maximum, des machines de très hautes performances sont

nécessaires pour pouvoir traiter d'immense corpus de textes et utiliser de très volumineux WE. Pour ainsi pouvoir exploiter les WE a de manière plus efficace

En perspectives, nous proposons de considérer entre autres les aspects suivants :

- Ajouter une étape de correction grammaticale automatique au début de l'étape du traitement du texte afin de corriger les erreurs de saisies et par la suite éviter les ambiguïtés.
- Le système est valide seulement pour une seule RM c'est-à-dire qu'il ne supporte pas une variété de RM. Nous supposons que l'ajout de cette fonctionnalité au système donnera encore mieux.
- Elaborer plus de justification à l'utilisateur/étudiant en retournant son score. Ceci peut être établit à l'aide des commentaires à propos de ses fautes et en retournant la RM.

- Considérer les entités nommées que nous avons ignorées dans le contexte de ce travail.

Bibliographies

1. **R. Siddiqui, C.J. Harrison and R. Siddiqi.** *"Improving Teaching and Learning through Automated Short Answer Marking," in IEEE Transactions on Learning Technologies.* 2010. pp. 237-249. Vol. 3.

2. **Wing, J.M.** *Computational thinking. Communication of the ACM, 49(3), 33-33. doi:10.1145/1118178.1118215.* 2006.

3. **Wachsmuth, H., Stein, B., and Engels, G.** *Information Extraction as a Filtering Task.* [ed.] Q. He and A. Iyengar. San Francisco : Proceeding of the Twenty-Second ACM International Conference On Information and Knowledge Management, CIKM'13, 2013. pp. 2049-2058.

4. *Constructing Efficient Information Extraction Pipeline.* [ed.] A. de vries, W. Fan, C. MacDonald, I. Ounis, and I. Ruthven B. Berendt. s.l. : Proceedings of the Twentieth ACM International Conference on Information and Knowledge Management, CIKM'11, 2011. pp. 2237-2240.

5. **Steven Burrows, Iryna Gurevych, and Benno Stein.** *The Eras and Trends of Automatic Short Answer Grading .* 01 25, 2015. International Journal of Artificial Intelligence in Education .

6. **Negre, Elsa.** *Compraison de textes: quelques approches.* 2013.

7. **Nielsen, R. D., Ward, W., Martin, J. H., and Palmer, M.** *Annotating Students' Understanding of Science Concepts.* [ed.] K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias N. Calzolari. Marrakech : Proceedings of the Sixth International Conference on Language Ressources and Evaluation, European Language Resources Association , 2008. pp. 1-8.

8. **Leacock, C. and Chodorow, M.** *C-rater: Automated Scoring of Short Answer Questions.* s.l. : Computers ans the Humanities, 37(4), 2003. pp. 389-405.

9. **Jordan, S.** *Investigating the Use of Short Free Text Questions in Online Assessment. Final project report,Centre for the Open Learning of Mathematics, Science, Computing and Technology, The Open University,Milton Keynes.* 2009.

10. **Abdallah A, Garoudja K.** *Mesures de similarité syntaxique pour un système d'évaluation automatique des réponses courtes : Application à la langue arabe.* 2018.

11. **S, Waterman. M.** *Idetifiatio of Coo Moleula "useuees Idetifiatio of Common Molecular Subseuee.* 1981. pp. 195-197.

12. **Fahmy, Wael Gomaa and Aly A.** *Article: A Survey of Text Similarity Approaches. International Journal of Computer Applications 68(13):13-18.* 2013.

13. **Mihalcea, R., Corley, C. & Strapparava, C.** *Corpus based and knowledge-based measures of text semantic similarity.* [ed.] Proceedings of the American Association for Artificial Intelligence. Boston : s.n., 2006.

14. **Islam, A., & Inkpen, D.** *Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data, 2(2).* 2008. pp. 1-25.

15. ***Efficient Estimation of Word Representations in. Vector Space. Tomas Mikolov. Google Inc., Mountain View, CA.***

16. Mikolov T, Kai Chen Gregory S. CorradoJeffrey A. Dean. ***Computing numeric representations of words in a high-dimensional space.***

17. M. Naili, A.H. Chaibi, dan H. H. Ben Ghezala. ***"Comparative Study of Word Embedding Methods in Topic Segmentation," Procedia Computer Science.*** **2017. pp. 340-349. Vol. 12.**

18. Caroline Gasperin, Pablo Gamallo, Alexandre Agustini, Gabriel Lopes, Vera de Lima. ***Using Syntactic Contexts for Measuring Word Similarity.*** **Porto, Lisbogne : s.n.**

19. Moschitti, Alessandro. ***Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees.*** **Department of Computer ScienceUniversity of Rome "Tor Vergata"Italy : s.n., 2006. pp. 318-329.**

20. Osgood, Charles. ***THE NATURE AND MEASUREMENT OF MEANING1.*** **1952. Vol. 49.**

21. David L.Waltz, Jordan B.Pollack. ***Massively parallel parsing: A strongly interactive model of natural language interpretation.*** **Coordinated Science Laboratory University of Illinois, USA : s.n., 1985. pp. 51-74. Vol. 9.**

22. Danilo Croce, Daniele Previtali. *Manifold Learning for the Semi-Supervised Induction.* Roma : s.n., 2010.

23. Jeffrey Pennington, Richard Socher, Christopher D. Manning. *GloVe: Global Vectors for Word Representation.* 2014.

24. M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, R. Zamparelli. *SemEval-2014 Task 1: Evaluation of Compositional Distributional.* Trento : s.n., 2014.

25. Mikolov, T., Chen, K., Corrado, G., Dean, J.:. *Efficient Estimation of Word Representations in Vector Space.* [ed.] ICLR: Proceeding of the International Conference on Learning Representations Workshop Track. Arizona : s.n., 2013. pp. 1301-3781.

26. Kong Joo Lee, Yong-Seok Choi, Jee Eun Kim. *Building an automated English sentence evaluation system for students learning English as a second language.* Daejeon : s.n.

27. Henniche. A N, Hannoufi. M H. *Les Word-Embedding pour l'évalutaion automatique des réponses courtes en apprentissage en ligne : Application à la langue arabe.* Université Saad Dahlab Blida, Soumaa : s.n., 2018.

28.] Lovins, Julie B. *Development of a stemming algorithm.Cambridge: MIT Information Processing Group, Electronic Systems Laboratory.* 1968.

29. [Online] [Cited: 08 15, 2019.] https://code.google.com/archive/p/word2vec/.

30. Abu Bakr Soliman, Kareem Eissa, Samhaa R. El-Beltagy. *AraVec: A set of arabic Word Embedding Models for use in Arabic NLP.* Precedia Computer Science. 2017. pp. 256-265.

31. [Online] [Cited: 08 15, 2019.] https://fasttext.cc/docs/en/support.html.

32. [Online] [Cited: 08 15, 2019.] http://mlg.ucd.ie/datasets/bbc.html.

33. [Online] [Cited: 08 15, 2019.] https://drive.google.com/file/d/0B2Mzhc7popBga2RkcWZNcjlRTGM/edit.

34. Michael Mohler, Reda Mihalcea. *Text-to-text Semantic Similarity for Automatic Short Answer Grading.* 2009.

35. [Online] [Cited: 08 15, 2019.] http://snowball.tartarus.org/algorithms/porter/stemmer.html.

36. [Online] [Cited: 08 15, 2019.] http://snowball.tartarus.org/algorithms/lovins/stemmer.html.

37. [Online] [Cited: 08 15, 2019.] https://www.nltk.org/_modules/nltk/stem/lancaster.html.

38. [Online] [Cited: 09 10, 2019.] https://github.com/anastaw/Meedan-Memory .

39. Bojan Furlan, Vuk Batanović, Boško Nikolić. *Semantic similarity of short texts in languages with a deficient natural language processing support, Decision Support Systems.* 2013. pp. 710-719. Vol. 55.

40. Nagoudi, El Moatez Billah & Schwab, Didier. *Semantic Similarity of Arabic Sentences with Word Embeddings.* 2017.

41. MacQueen, J. B. *« Some Methods for classification and Analysis of Multivariate Observations » Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.* 1967. pp. 281-297.

42. [Online] [Cited: 09 10, 2019.] https://github.com/llimllib/ckmeans.

43. Altman, N. S. *"An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3).* 1992. pp. 175-185.

44. Cohen, Jacob. *Statistical power analysis for the behavioral sciences (2nd ed.).* 1988.

45. Greene, William H. *Econométrie, Paris, Pearson Education, 5e éd. (ISBN 978-2-7440-7097-6).* 2005.

46. S, Abdellaoui. *Mesures de similarité sémantique pour un système d'évaluation automatique des réponses courtes : Application à la langue anglaise.* Blida : s.n., 2019.

47. [Online] [Cited: 04 12, 2019.] https://stanfordnlp.github.io/CoreNLP/ .

48. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. *Distributed Representation of Words and Phrases and their Compositionality. In:.* s.l. : NIPS: Proceedings of Neural Information Processing Systems , 2013. pp. 3111-3119.

49. [Online] [Cited: 09 10, 2019.] https://link.springer.com/chapter/10.1007/978-90-481-8847-5_10.

50. ***Bloom Taxonomie: A taxonomy for learning, teaching, and assessing : a revision of Bloom's taxonomy of educational objectives.*** **New York : Longman : s.n., 2001.**

51. Valenti, S., Neri, F., and Cucchiarelli, A. ***An Overview of Current Research on Automated Essay Grading Journal of Information Technology Education, 2.*** **2003. pp. 319-330.**

52. Pérez-Marín, D., Pascual-Nieto, I., and Rodríguez, P. ***Computer-Assisted Assessment of Free-Text Answers. The Knowledge Engineering Review, 24(4).*** **2009. pp. 353-374.**

53. Burstein, J., Kaplan, R., Wolff, S., and Lu, C. ***Using Lexical Semantic Techniques to Classify Free Responses.*** **[ed.] E. Viegas. Santa Cruz : Proceedings of the ACL SIGLEX Workshop on Breadth and Depth of Semantic Lexicons, 1996. pp. 20-29.**

54. Sukkarieh, J.Z. and Blackmore, J. ***c-rater: Automatic Content Scoring for Short Constructed Reponses.*** **[ed.] H. C. Lane and H. w. Guesgen. Sanibel Island : Proceedings of the Twenty-Second International Conference of the Florida Artificial Intelligence Research Society, 2009. pp. 290-295.**

55. Sukkarieh, J. Z. and Stoyanchev, S. ***Automating Model Bulding in c-rater.*** **[ed.] Callison_Burch and F. M. Zanzotto. s.l. : Proceedings of the First ACL/IJCNLP Worlshop on Applied Textual Inference TextInfer '09, 2009. pp. 61-69.**

56. Myroslava Dzikovska, Peter Bell, Amy Isard, and Johanna D. Moore. ***Evaluating language understanding accuracy with respect to objective outcomes in a dialogue system. In EACL.*** **[ed.] The Association for Computer Linguistics. 2012. pp. 471-481.**

57. Mitchell, T., Broomhead, P, and Aldridge, N. ***Towards Robust Computerised Marking of Free Text Responses.*** **s.l. : Proceedings of the Sixth Computer Assisted Assessment Conference, 2002.**

58. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. ***Indexing by latent semantic analysis. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 41(6) :391407.*** **1990.**

59. Bukai, O., Pokorny, R., and Haynes, J. ***An Automated Short-Free-Text Scoring System: Development and Assessment.*** **[ed.] National Training and Simulation Association. s.l. : Proceedings of the Twentieth Interservice/Industry Training, Simulation, and Education Conference, 2006. pp. 1-11.**

60. Bailey, S. and Meurers, D. [ed.] J. Burstein, and R. De Felice, editors J. Tetreault. Columbus : Diagnosing Meaning Errors in Short Answers to Reading Comprehension Questions. Association for Computational Linguistics, 2008.

61. Dzikovska, M. O., Nielsen, R. D., Brew, C., Leacock, C., Giampiccolo, D., Bentivogli, L., Clark, P., Dagan, I., and Dang, H. T. *SemEval-2013 Task 7: The Joint Student Response Analysis and Eighth Recognizing Textual Entailment Challenge.* [ed.] T. Baldwin, and M. Baroni M. Diab. Atlanta : s.n., 2013. pp. 1-12.

62. Prettenhofer, P. and Stein, B. *Cross-Lingual Adaptation using Structural Correspondence Learning. Transactions on Intelligent Systems and Technology, 3, 13:1–13:22.* 2011.

63. Hirst, Alexander Budanitsky and Graeme. *Semantic Distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures.* Toronto : s.n., 2001.

64. Allan M. Collins, M. Ross Quillian. *Retrieval time from semantic memory.* 1969. pp. 240-247. Vol. 8.