République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Systèmes d'information

Thème

Techniques d'intégration de mots « Word Embedding » pour l'indexation sémantique du contenu de transcription des ressources parlées.

Présenté Par :
DJEDDAI
Imad

Devant le jury :

Mr. Djeddai. A	*MCB*	*Univérsité Larbi Tébessi*	*Président*
Mr. Khediri. S	*MAA*	*Univérsité Larbi Tébessi*	*Examinateur*
Mr. Bendib. I	*MCB*	*Univérsité Larbi Tébessi*	*Encadreur*

Date de soutenance : 23 Juin 2019

Résumé

L'utilisation des systèmes informatiques intelligents pour l'accès aux ressources numériques est incontournable dans notre quotidien. En effet, ces systèmes sont utilisés en large échelle comme les Web ainsi que dans les systèmes fermés. Cependant, l'extension de ces systèmes vers l'exploitation des ressources numériques complexes tels que l'image, parole et multimédia est fortement sollicitée. Dans ce contexte, nous présentons dans ce mémoire une étude pour l'intégration des nouvelles techniques relatives aux indexations sémantiques des ressources numériques pour les systèmes de recherche d'information. Dans ce travail, nous avons étudié les possibilités d'intégration des techniques de « Deep Learning » et « Word Embedding » dans le processus d'indexation sémantique pour les résultats des transcriptions automatiques du contenu des ressources parlées. Dans ce cadre, nous avons présenté principalement une contribution pour l'utilisation de la représentation vectorielle sémantique « Word2Vec » avec ces approches de calculs de similarités dans le processus d'indexation sémantique basée sur un modèle « Deep Learning ». Cependant, pour la validation nous avons évalué cette approche avec un classifieur convolutionnel « CNN » dans le but de mesurer l'impact de la stratégie d'indexation proposée. Les expérimentations sont effectuées sur un extrait du corpus TED des ressources et les résultats obtenus à l'aide de l'algorithme « word2vec » sont encourageants.

Les Mots clés *: Indexation sémantique, Word Embedding, Apprentissage profond, RNN, Word2vec, CNN.*

Abstract

In our daily lives the use of intelligent computer systems to access digital resources is essential. Indeed, these systems are used on a large scale like the Web as well as in closed systems. However, the extension of these systems to the exploitation of complex digital resources such as images, speech and multimedia is in high demand. In this context, we present in this thesis a study for the integration of new techniques relating to semantic indexations of digital resources for information retrieval systems. In this work, we investigated the possibilities of integrating "Deep Learning" and "Word Embedding" techniques into the semantic indexing process for the results of automatic transcriptions of the speech content resources. In this context, we mainly presented a contribution for the use of the semantic vector representation "Word2Vec" with these approaches of similarity calculations in the semantic indexing process based on a "Deep Learning" model. However, for the validation we evaluated this approach with a convolutional classifier "CNN" in order to measure the impact of the proposed indexation strategy. The experiments are performed on an extract from the TED corpus of resources and the results obtained using the "word2vec" algorithm are encouraging.

Keywords: *semantic indexing, Word Embedding, Deep Learning, RNN, Word2Vec, CNN.*

مـلـخص

يعد استخدام أنظمة الكمبيوتر الذكية في حياتنا اليومية للوصول إلى الموارد الرقمية أمرًا ضروريًا. حيث، يتم استخدام هذه الأنظمة على نطاق واسع مثل الويب وكذلك في الأنظمة المغلقة. ومع ذلك، فإن امتداد هذه الأنظمة نحو استغلال الموارد الرقمية المعقدة مثل الصور والكلام والوسائط المتعددة هو أمر مطلوب بشدة. في هذا السياق، نقدم في هذه الرسالة دراسة لدمج التقنيات الجديدة المتعلقة بالفهرسة الدلالية للموارد الرقمية في نظام البحث المعلوماتي. في هذا العمل، بحثنا في إمكانيات دمج تقنيات التعليم العميق وتضمين الكلمات في عملية الفهرسة الدلالية لنتائج نصوص محتوى الكلام التلقائي. في هذا الإطار، قدمنا أساسًا مساهمة في استخدام تمثيل المتجه الدلالي "Word2Vec" مع نُهج حساب التشابه بين الكلمات هذه في عملية الفهرسة الدلالية استنادًا إلى نموذج "التعلم العميق". ومع ذلك، للتحقق من صحة قمنا بتقييم هذا النهج مع مصنف تلافيفي "CNN" من أجل قياس تأثير استراتيجية الفهرسة المقترحة. يتم إجراء التجارب على مقتطف من مجموعة موارد TED والنتائج التي تم الحصول عليها باستخدام خوارزمية "word2vec" مشجعة.

<u>الكلمات المفتاحية</u>: الفهرسة الدلالية، تضمين الكلمات، التعلم العميق، RNN، Word2Vec ، CNN.

Remerciments

*Tout d'abord, longue vie à « **Allah** » qui nous a guidé sur le droit chemin tout au long du travail et nous a inspiré les bons pas et les justes reflexes. Sans sa miséricorde, ce travail n'aurait pas abouti.*

*L'encadrement scientifique de ce travail a été assuré par **Dr. Bendib Issam**, maitre de conférences classe B à la faculté des Sciences Exactes et des Sciences de la Nature et de la Vie, Université Tébessa. Nous tenons vivement à lui exprimer nos profonde reconnaissances gratitude pour sa disponibilité, sa patience, sa compréhension, ses qualités humaines et ses intérêts portés pour notre sujet de travail. Nous le remercions de nous avoir fait confiance et d'avoir été présent aussi souvent que possible malgré ses tâches pédagogiques. Son soutien permanent et son dynamisme nous ont permis d'avancer plus loin dans notre travail.*

Nos remerciements vont aussi à

*Dr. **Djeddai. A,** d'avoir ménagé son temps pour présider ce jury*

*Dr. **Khediri. S,** pour avoir bien voulu siéger dans ce jury afin d'examiner et critiquer ce mémoire et nous éclairer par ces précieux conseils.*

*Dr. **Bendib. I***

Aucun remerciement ne saurait exprimer notre respect et considération pour les orientations que vous avez consenties pour notre étude de l'Université.

*Mr. **Gattel. A**, Mr. **Gahmouss. A,** Mr. **Laimech. A**, Mr. **Hmidane. F**, Mme. **Bourougaa. S**, tous les enseignants qui nous ont fait former durant ces 5 années merci pour votre encouragement et gentillesse.*

Tous nos amis pour leur solidarité

La meilleure équipe, l'équipe de l'informatique

A nos parents

Pour l'enfance merveilleuse qu'ils nous ont offerte ainsi que pour leurs encouragements.

Pour leurs soutiens et leurs aides.

Avec tout notre amour.

A tous mes amis

Pour leurs bonnes humeurs, leurs gentillesses et pour tous nos fous rires partagés.

Pour tout ce qu'ils nous ont appris

Merci à tous ceux qui, d'une manière ou d'une autre qui ont contribué à la réalisation de ce travail, et que nous ne pouvons citer individuellement.

DEDICACE

Je dédie ce travail

A mes parents.

A mes sœurs : Ibtissam et Assia.

A mes frères : Samir, Fares et Abdelaziz.

A toutes mes amies : Aimen, Mourad, Rafik, khairi,

Abdessami, Ali, Mouhamed, Zine, Saad et Zaki.

A tous ceux qui m'ont aidé dans mes études.

A tous mes proches, et tous ceux qui m'aiment.

Imad

Table des matières

Liste de Figures

Liste de Tableaux

Introduction Générale

1. Motivation

Le développement accru des systèmes informatiques intelligents dans le domaine d'intelligence artificielle a permis l'intégration de ces systèmes dans la vie quotidienne. En effet, les supports numériques s'intègrent de plus en plus dans les différentes plateformes et systèmes informatiques dans les différents domaines. Entre temps, l'avènement du Web et le partage de ressources entre les différentes communautés éparpillées dans le monde ont encouragé les équipes de recherches de s'engager sur le développement des techniques et outils intelligents pour accroitre les performances de ces systèmes.

Entre autres, nous trouvons que les systèmes de recherche d'information prennent une grande part de ces efforts de recherches. En effet, la disponibilité des ressources numériques autres que le texte, telles que l'image, la parole et le multimédia encouragent la recherche d'intégration de ces ressources dans les systèmes informatiques. À cet effet, nous trouvons plusieurs travaux de recherche sur les techniques d'indexation utilisée pour accroitre les performances des systèmes de recherche d'information.

D'autre part, nous trouvons l'évolution et les résultats obtenus dans les techniques utilisées pour le développement des modèles de représentation de connaissances sont très encourageants. En plus nous trouvons l'intégration d'autre support de connaissances autre que les ontologies telles que les méthodes d'apprentissage statistique et profond deviennent des pistes de recherches très encourageantes. Dans ce contexte, que nous proposons dans ce mémoire une étude sur l'intégration de ces techniques dans le processus de segmentation.

2. Problématiques

Les performances des systèmes de recherche d'informations sont liées étroitement par les modèles de représentations de ressources et les fonctions de correspondances utilisées ainsi que les stratégies de segmentation déployées. En effet, les performances enregistrées sur les ressources textuelles sont très encourageantes de point de vue syntaxique. Aussi, nous trouvons aussi une évolution importante de ces systèmes pour le traitement sémantique du contenu de ces ressources et les techniques de modélisation des connaissances. Cependant, ces techniques sont généralement utilisées pour le développement des modèles de connaissance pour des domaines spécifiques.

En revanche, la création des systèmes de recherche d'information à base d'indexation sémantique nécessite le recours vers des modèles de représentation de connaissance plus généralisée. Dans ce contexte, nous trouvons les travaux qui utilisent l'ontologie « *WordNet* » comme un support de connaissances. Notons aussi que la version la plus riche de « *WordNet* » est pour la langue anglaise. L'enrichissement du « WordNet » pour les autres langages est l'un des pistes de recherche en cours. Dans ce contexte, nous avons opté pour l'étude d'utilisation d'autre support de connaissance que « WordNet ».

D'autre part, le développement des systèmes de recherches d'information pour des ressources plus complexes que les ressources textuelles nécessite l'intégration des techniques et méthodes relatives aux types de ressources à traiter. En effet, le traitement du volume important des ressources numériques multimédias existant par les systèmes de recherche d'information est incontournable. Aussi, nous trouvons que la primitive parole dans ces ressources multimédias est largement utilisée. D'où, la nécessité d'intégrer les techniques de traitement de parole pour les systèmes de recherche d'information.

En revanche, les systèmes de reconnaissances automatiques de la parole sont des solutions utilisées pour le passage de la primitive parole vers la primitive texte afin de les rendre accessibles et exploitables par les systèmes de recherche d'information. Cependant, la qualité de transcriptions automatiques obtenues des ressources parlées est déterminante pour les performances des stratégies d'indexations déployées. Cependant, malgré les performances enregistrées dans ces systèmes de reconnaissances automatiques de la parole. Ils restent incapables de gérer les problèmes liés aux reconnaisses automatique de la parole telle que : les mots techniques, les mots étranges et les langages moins dotés. À cet effet, le recours vers les méthodes d'apprentissage statistique lors du processus d'indexation sémantique s'avère très encourageant.

3. Contribution à l'indexation sémantique

Notre contribution dans ce manuscrit consiste à étudier la possibilité d'intégrer les techniques d'apprentissage profond pour la construction d'un modèle de représentation de connaissance qui sera utilisé pour l'indexation sémantique du contenu des ressources numériques. En effet, l'indexation classique à base du modèle vectoriel permet le traitement des mots et indexes selon leurs fréquences d'apparitions. Cependant, l'indexation sémantique permet de détecter l'aspect sémantique intégré entre les termes des documents. Dans ce contexte, que nous proposons une stratégie d'indexation sémantique qui permet de détecter de façon automatique les similarités possibles entre les termes importants du contenu des ressources numériques. En effet, nous utilisons la primitive texte obtenu par la transcription automatique du contenu des ressources parlées. Afin d'alléger l'impact de la qualité de la transcription automatique sur les résultats de segmentation, nous recourons aux modèles « *Word Embedding* » pour éliminer les erreurs engendrées par les systèmes de reconnaissance automatique.

Notre stratégie de segmentation sémantique proposée est développée autour de trois axes suivants :

- La construction du modèle de représentation avec l'utilisation de l'algorithme « *Word2Vec* » comme une implémentation de la technique « *Word Embedding* » pour la construction d'un modèle de représentation vectorielle des termes des ressources à indexer.
- L'exploitation de modèle avec l'utilisation les fonctionnalités disponibles dans les modèles « *Word Embedding* » pour mesurer les similarités sémantiques existantes entre les termes susceptibles d'indexation.

- L'évaluation du modèle par la réalisation d'un classifier CNN avec la technique de « *Deep Learning* » pour évaluer les résultats d'indexation sémantique obtenus par notre modèle vectoriel définit par « *Word Embedding* ». En effet, l'utilisation de classifier CNN est fondée par la représentation matricielle (vectoriel) uniforme obtenue par l'algorithme « Word2Vec » pour chaque mot ou index retenu.

4. L'organisation du manuscrit

Ce manuscrit est composé de quatre chapitres et une introduction générale. Le premier chapitre introduit brièvement les concepts relatifs aux techniques d'indexation, les techniques de représentation de l'information et des connaissances ainsi que les particularités des caractéristiques et structure des documents parlés.

Dans le deuxième chapitre, nous définissons les grandes lignes des techniques utilisées pour le domaine d'apprentissage profond « *Deep Learning* » et « *Word Embedding* » ainsi que leurs implémentations comme « Word2vect ». Ainsi, nous avons présenté une description sur les bibliothèques et les environnements largement utilisés dans ce domaine.

Le troisième chapitre est consacré à notre contribution par l'étude des techniques d'intégration du modèle « *Word Embedding* » avec l'algorithme « Word2vec » pour l'indexation sémantique du contenu des transcriptions des ressources parlées. Aussi, nous présentons dans ce chapitre l'architecture de classifier « CNN » réalisée pour l'évaluation de la qualité de la stratégie d'indexation sémantique déployée.

Ainsi, dans le quatrième chapitre nous présentons les stratégies utilisées pour l'implémentation et la validation de notre approche. Nous définissions les ressources parlées utilisées ainsi que les différents environnements et routines développés et les résultats obtenus avec les analyses et les synthèses appropriées. Enfin nous clôturons ce manuscrit par une conclusion sur les travaux réalisés ainsi que nos perspectives futures.

Chapitre 1

L’indexation sémantique et les supports de connaissances

1. Introduction

Le domaine de recherche d'information est un axe de recherche multidisciplinaire qui utilise plusieurs disciplines et techniques de traitement d'information. Les évolutions progressives de ces techniques dans ces dernières années ont permis de tracer plusieurs pistes de recherches dans cette discipline. Parmi eux, nous trouvons les travaux de recherches liés aux systèmes d'information automatisés, systèmes de gestion d'information, systèmes de gestion de bases de données, les systèmes de support à la décision et les systèmes de recherche d'information. Les évolutions récentes des technologies d'information ont modifié les types de ressources et multiplié les volumes de données. Ainsi, elles introduisent l'applications de techniques la gestion électronique de documents et les systèmes d'information multimédias.

En effet, les systèmes de recherche d'information ont pris une grande part dans l'univers des systèmes informatique et cela due par la nécessitée étroite de la recherche et l'accès aux informations numérique éparpillés dans les réseaux informatiques. Entre autres, la nécessité de développer ou d'améliorer les techniques d'accès et recherche de ces ressources numérique sont des pistes de recherches très sollicités vue l'impact direct de ces recherches dans l'exploitation et l'utilisation des systèmes informatiques.

Dans ce contexte, nous présentons dans ce chapitre les techniques liées aux accès et recherche d'information numérique telles que l'indexation et la représentation et modélisation des connaissances. Ces dernières sont des supports techniques utilisés pour le développement des systèmes de recherche d'information.

2. Le processus d'indexation

2.1. Principes de l'indexation

Nous trouvons dans les systèmes de recherche d'information que les documents sont définis comme des supports d'informations auto-explicatives. Afin d'exploiter ces ressources, nous trouvons dans les littératures des techniques utilisées pour trouver une définition précise pour les termes constituant ces documents afin de maximiser l'expressivité sémantique des termes. L'indexation consiste à trouver les termes qui représente synthétiquement le contenu sémantique des documents, ces termes sont appelés « index ».

Techniquement, Les index sont représentés dans un modèle et langage spécifique. Le formalisme qui permet de représenter syntaxiquement un index est appelé le langage d'indexation ainsi que les techniques utilisées pour représenter l'expressivité des index est appelé modèles de représentation. La complexité de ces formalismes est liée étroitement par les modèles mathématiques utilisés ainsi que la structure des ressources à traitées [Bendib I, 2018].

Cependant, afin de concevoir un système de recherche performant, il est important que les index utilisés représentent et reflètent au maximum le contenu des ressources de la collection originale. Indexer un document c'est d'extraire ses termes représentatifs afin de générer la

liste des termes d'indexation susceptibles et construire avec ces indexes une liste de références de collection.

2.2. Stratégies d'indexation

En effet le choix termes d'indexation n'est pas implicite car nous trouvons parfois que les termes interdépendants sont sensibles au contexte. Pour cela, pour trouver ces termes ou indexes, il faut qu'on exécute une série d'étapes de prétraitements. Nous résumons ces étapes dans les sections suivantes.

2.2.1. Le prétraitement lexical

L'objectif de cette étape est l'extraction des termes à partir du contenu d'un document textuel. Cette extraction est réalisée après quelques prétraitements classiques comme l'élimination des signes de ponctuations et la casse. Entre temps, pour détecter les termes représentatifs des ressources textuelles, nous trouvons plusieurs techniques qui peuvent être utilisées comme l'élimination des mot vide ou les mots fonctionnels de la langue qui sont appelés souvent « stop word »[Luhn H.P,1959]. En pratique, nous utilisons souvent le concept anti dictionnaire pour représenter cette liste de mots. Parmi eux, nous citons comme exemple : «Snowball stop word[1] » ,« Terrier stop word[2]»,« XPO6[3] »

2.2.2. Indexation par radicaux « Stemming »

Le processus d'indexation par radicaux permet de représenter plusieurs variantes d'un mot sous une forme unique appelée racine ou radical. Nous trouvons dans la littérature que la différence morphologique entre la racine et radical est faite : la racine est la forme abstraite servant de base de représentation de tous les radicaux communs. En effet le radical d'un mot est une simple réduction du nombre de lettres de ce mot, et celui-ci peut différer que la racine morphologique correcte. Par exemple, le mot « computation » peut être représenté par plusieurs radicaux « computa », « comput », « compu », sa racine linguistiquement correcte étant « compute ».

En revanche, Les algorithmes qui permettent de transformer un mot vers son radical sont appelés les algorithmes de radicalisation « Stemming ». En effet, ces algorithmes de radicalisation peuvent être linguistiques, comme par exemple l'algorithme de « Porter » [Porter M.F, 1980]

[1] http://snowball.tartarus.org/algorithms/english/stop.txt
[2] https://github.com/RxNLP/text-mining-and-nlp-apis/blob/master/terrier-stop-word-list.txt
[3] http://xpo6.com/list-of-english-stop-words/

2.2.3. Indexation par lemmes « Lemmatization »

L'indexation par lemmes est une méthode plus fine qui se concentre exclusivement sur l'impact des catégories grammaticales sur les termes, en amenant tous les mots vers leur lemme. L'avantage de cette méthode par rapport à celle basée sur les radicaux concerne surtout les formes courtes et/ou irrégulières, et les verbes. Notons aussi que cette méthode utilise les concepts du lexique morphologique et nécessite une opération préalable de catégorisation des unités lexicales comme par exemple le terme « détectable » sera lemmatisé en « détecter » parce qu'il est son adjectif. A cet effet, plusieurs types d'erreurs peuvent être générées dans ce processus, notamment sur les mots inconnus du lexique de référence utilisé et les unités ambiguës sur le plan catégoriel.

2.2.4. La pondération des termes

La pondération des termes dans le processus d'indexation permet d'associer un poids de représentativité $w_{i,j}$ pour chaque terme t_j d'un document d_i. De manière générale, les formules de pondération utilisées sont basées sur la combinaison d'un facteur de pondération local quantifiant, la représentativité locale du terme dans le document, et d'un facteur de pondération global quantifiant la représentativité globale du terme vis-à-vis de la collection de documents. Les différentes approches et techniques de pondération sont détaillées dans les travaux effectués par Salton [Salton & al, 1993].

Dans ce contexte, plusieurs formules existent, nous citons à titre d'exemple la mesure la plus utilisée *tf-idf*. Cette mesure représente une bonne approximation de l'importance d'un terme dans un document par rapport au corpus cible. Elle est particulièrement utilisée dans des corpus de documents de tailles intermédiaires. Elle est calculée comme suit :

$$w_{i,j} = \frac{tf_{i,j}}{df_j} = tf_{i,j} \times \frac{1}{df_j} = tf_{i,j} \times idf_j \quad [Salton\ \&al, 1993] \qquad (01)$$

Où :

$tf_{i,j}$: est la fréquence d'occurrences du terme t_j dans le document d_i

df_j : est la fréquence documentaire du terme t_j

idf : est la fréquence documentaire inverse

2.3. Les approches d'indexation

L'indexation des ressources est une étape principale pour l'exploitation des ressources d'information disponibles dans les systèmes informatiques ; figure 1.1. Cette tâche utilise plusieurs approches pour trouver la définition des index qui seront considérés comme des termes représentatifs du contenu des ressources. Techniquement, l'indexation peut être manuelle, automatique ou semi-automatique, nous trouvons trois formes d'indexation :

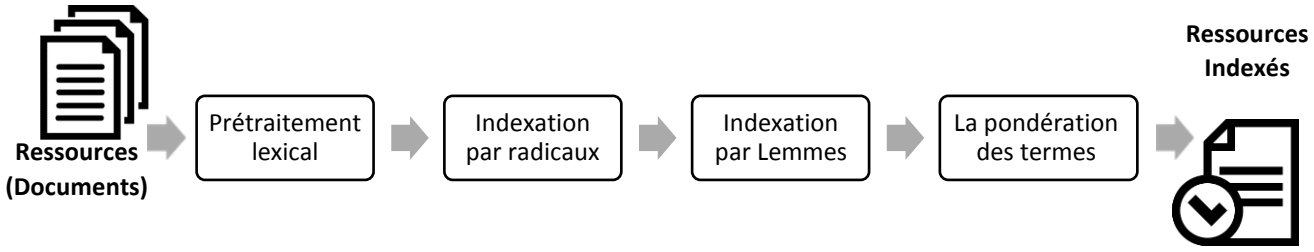

FIGURE 1.1 : LES ÉTAPES D'INDEXATION D'UN DOCUMENT.

2.3.1. L'indexation manuelle

Ce mode d'indexation est pratiqué généralement par les professionnels des bibliothèques et les experts des domaines en se basant sur des outils linguistiques spécifiques tels que les thésaurus[4] et les répertoires. Il s'agit donc d'une indexation à la fois humaine, manuelle contrôlée par des langages documentaires.

Nous trouvons dans les littératures des travaux basés sur une indexation manuelle limités à des termes simples choisit manuellement fournit des bonnes performances comparant à celles sur les listes de vocabulaire contrôlé « *Thésaurus* ». En revanche, le nombre important de documents stockés sur les supports électroniques ne permet pas de recourir aux telles techniques d'indexations, ce qui impose de passer vers les méthodes automatiques.

2.3.2. L'indexation automatique

L'indexation automatique repose sur l'extraction a base des calculs statistiques les termes d'indexation des contenus des documents eux-mêmes. Ce type d'indexation est composée de deux étapes : la recherche des termes caractérisant le contenu et l'évaluation du pouvoir de caractérisation de ces termes. Elle est caractérisée par :

- Le choix du type d'unités d'indexation : radical, mot simple, groupe de mots.
- La définition des règles d'équivalence entre termes issus des documents et termes d'indexation (radicalisation, lemmatisation, troncature, etc.),

[4] https://fr.wikipedia.org/wiki/Thésaurus

- Le principe de sélection des termes représentatifs du document et ceux qui ne le sont pas, en fonction du contenu du document (termes d'indexation),

Le résultat de ce type d'indexation est un ensemble de couples (terme d'indexation, poids) associés à chaque document. Ce type d'indexation à l'avantage de sa facilité d'implémentation et d'intégration dans les systèmes automatiques [Bendib I, 2018]. Cependant, cette approche doit confronter quelques problèmes comme :

- La correspondance entre les termes d'indexation et les unités sémantiques.
- Les métriques d'équivalences utilisées entre les termes.
- Les ambigüités entre les termes engendrés par le traitement automatique.

2.3.3. L'indexation sémantique linguistique

Cette stratégie permette l'amélioration de la précision des recherches, en réduisant le nombre de fausses alertes. L'information sémantique est extraite à partir d'un traitement du langage plutôt que de traiter chaque mot comme une entité statistique indépendante. La sortie la plus simple de ce processus est la génération des indexes sous formes de phrases pour les ressources. Aussi, une analyse plus complexe génère une représentation thématique des ressources plutôt que des phrases. L'indexation a base syntaxiques et sémantiques générées par des algorithmes de traitement du langage naturel améliorent la spécification de l'indexation en fournissent un autre niveau de désambiguïsation tel que les distances sémantiques. Le traitement du langage naturel peut également combiner les concepts par d' agrégations vers des concepts de niveau supérieur parfois appelés représentations thématiques. [Kowalski G, 2010].

2.4. Techniques d'indexation sémantique linguistique

2.4.1. Analyse sémantique latente « LSA »

C'est une approche qui permet de mesurer la similarité entre les textes. Il a été prouvé que la *« LSA »* saisissait des parties significatives de la signification non seulement de mots individuels, mais également de tous les passages tels que phrases, paragraphes et essais brefs. « *LSA* » est né du problème de trouver des documents pertinents à partir de mots de recherche. « *LSA* » est conçu pour surmonter le problème fondamental de la comparaison de mots pour trouver des documents pertinents en comparant les concepts ou les significations derrière les mots pour récupérer des documents [Rajaraman A & al, 2011].

2.4.2. Similarité de phrase « *STATIS* »

Les techniques de phrase Similarité « *STATIS* » est utilisé pour calculer efficacement la similarité entre des textes très courts ou des phrases. « *STATIS* » s'est avéré être l'une des meilleures techniques de précision de recherche de pages et d'images sur le Web [Yuhua L. & al, 2006]. Dans la récupération de page Web, « *STATIS* » utilise des titres pour représenter les documents dans le processus de recherche de page et dans la récupération d'image, le texte court entourant les images est utilisé à la place du document entier dans lequel l'image est incorporée [Yuhua L. & al, 2006]. La figure 1.2 montre comment deux phrases différentes sont calculées Dans l'approche « *STATIS* », les deux phrases sont déconstruites et placées dans des vecteurs sémantiques bruts et des vecteurs d'ordre séparés, organisés par une base de données lexicale. Les vecteurs d'ordre gardent une trace de l'ordre dans lequel les mots de chaque phrase se produisent. Les mots significatifs de chaque vecteur sémantique sont ensuite pondérés en utilisant un contenu d'information dérivé d'un corpus de texte. Enfin, la similarité des phrases est calculée en comparant les vecteurs les uns aux autres [Yuhua L. & al, 2006].

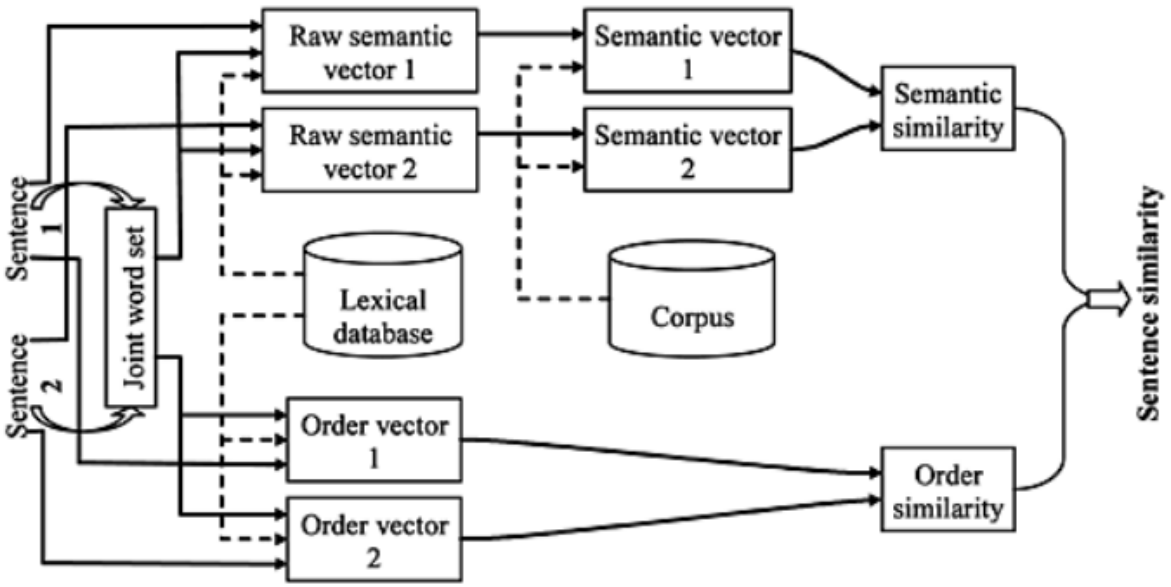

FIGURE 1.2 : DIAGRAMME DE CALCUL DE LA SIMILARITÉ DES PHRASES, IMAGE DE [YUHUA L. & AL, 2006].

2.5. L'évaluation des performances de l'indexation

Les performances de la stratégie de segmentation sont évaluées selon les performances des systèmes de recherche d'information qui les utilisent. Les critères utilisés pour l'évaluation de performance de ces systèmes les plus populaires sont basées sur des mesures statistiques relatives au nombre de ressource restituées par rapport aux requêtes utilisateurs. Dans le tableau 1.1 nous présentons quelques métriques les plus souvent utilisées avec une brève description ainsi que leurs règles de calculs.

Mesure	Description	Définition
Rappel	Rappel exact par rapport à l'ensemble des documents retrouvés. Le rappel mesure la capacité du système à restituer l'ensemble des documents pertinents (en lien avec le silence documentaire).	$R = \frac{Nb\ de\ Doc\ pertinents\ retrouvés}{Nb\ de\ Doc\ Trouvés}$
Precision	Précision moyenne non interpolée par rapport à l'ensemble des documents pertinents. Mesure la capacité du système à ne restituer que des documents pertinents	$P = \frac{Nb\ de\ Doc\ pertinents\ retrouvés}{Nb\ de\ Doc\ pertinents}$
MAP	Elle calcule la précision et le rappel à chaque position dans la séquence classée de documents pour les systèmes qui renvoient une séquence classée de documents	$MAP = \frac{1}{\|R\|}\sum_{i=1}^{n} precision(i).relevance(i)$
F1-Score	Le score F1 est la moyenne harmonique de la précision et du rappel, où un score F1 atteint sa meilleure valeur à 1 (précision parfaite et rappel) et le pire à 0.	$F1 = 2 * \frac{P \times R}{P + R}$

TABLEAU 1.1 : LES CRITERES D'ÉVALUATION LES PLUS POPULAIRES DANS LES SRI.

3. Les documents parlés :

3.1. Les caractéristiques des documents parlés

Les documents parlés partage plusieurs caractéristiques avec les documents textes classiques. Cependant, leurs contenus est souvent complexe et même incomplet et dépend étroitement par le style de parole utilisé tel que parole continu, parole spontanées, discours, etc. De point de vue structurel, il existe une grande différence entre la structure des documents texte et celles parlés qui soulèvent de nouveaux challenges qui doivent être abordées et traitées dans le processus de recherche d'information.

Parmi ces challenges, nous citons que le flux parlé est un support plus riche et plus expressif que le texte. En plus, il contient plus d'informations que les mots parlés comme : les caractéristiques des locuteurs, style de parole, dialecte, …etc. Avec telle ressource parlée, des informations supplémentaires telles que l'identité de la langue ou dialecte parlée, l'identité du locuteur, l'humeur ou le ton des locuteurs sont capturées parallèlement avec les mots parlés. Cependant, ces informations supplémentaires peuvent être utiles dans le développement des systèmes d'indexation et de recherche d'informations et elles offrent de nouvelles voies d'exploitation et d'application dans différentes contextes [Bendib I, 2018].

3.2. Les documents parlés versus documents textes

Il est important de chercher des méthodes et démarches pour l'extraction et la représentation du contenu des document parlés sous une forme adéquate pour éventuelle systèmes d'indexation et de recherche dans ces flux parlés. Bien que ces objectives sont similaire à celles visées pour les documents de textes, le passage du modalité texte vers la modalité parole engendre une nouvelle dimension de complexité et d'incertitude. En effet, la méthode ou l'approche sollicité doit confronter un ensemble de défis comme : la capacité de traiter l'aspect multi locuteurs, l'impact des bruits dans les flux parlés et le traitement des langages a large vocabulaire ainsi que les langages moins dotés.

En effet, la plupart des méthodes d'indexation et de recherche qui ont été développés pour les documents texte suppose implicitement que les transcriptions se génèrent sans erreur. Avec le texte, les mots dans les documents sont supposés être connus avec une certitude élevée. Par conséquent, il n'y a pas de mécanisme explicite dans les modèles pour le traitement des erreurs dans la représentation du document. Cependant, avec la parole, il n'y a pas actuellement des méthodes de transcription automatique parfaite et surtout pour certains styles de parole et il y aura implicitement des erreurs de transcriptions générées par les systèmes de reconnaissances automatique de la parole.

Dans ce contexte, l'utilisation des dictionnaires linguistiques pour la vérification des résultats de transcription est largement sollicitée. Entre autres, l'aspect sémantique est une nécessité majeure pour surmonter les problèmes de transcription. A cet effet, l'utilisation des modèles de représentation sémantique comme les thesaurus ou les ontologies avec la combinaison de mots à leurs radicaux pour l'amélioration de la qualité de la transcription automatique sont des approches prometteuses.

4. La Représentation des connaissances

Les tendances actuelles pour la représentation des connaissances se déroulent autour des ontologies. Ils ont la puissance de modélisation de l'univers en tenant compte des concepts avec ces relations entre eux. Les problèmes rencontrés auparavant dans les systèmes de recherche d'information de point de vu le traitement de l'aspect sémantique sont dégagés. Cependant d'autres difficultés résident toujours dans les systèmes de recherche d'information comme les ambigüités des mots lors de la création des indexes fiables et discriminant ou la formulation des besoins utilisateur [Hubert G, 2009].

4.1. Les ontologies

Les ontologies offrent une modélisation des connaissances d'un domaine basée sur une hiérarchie des concepts et termes d'un domaine. Ils ont souvent utilisé dans les Systèmes de Recherche d'Information (SRI) dans les taches d'indexation du contenu des ressources et l'enrichissement du langage d'interrogation requête. En effet, ils permettent notamment de surmonter les problèmes d'ambiguïtés lexicales et sémantiques du langage des ressources dans les SRI classiques [Sy M.F, 2012]. Actuellement, les ontologies ont pris une grande part

dans le domaine de représentation de connaissance. La figure 1.3, présente une représentation simplifiée des concepts d'une ontologie.

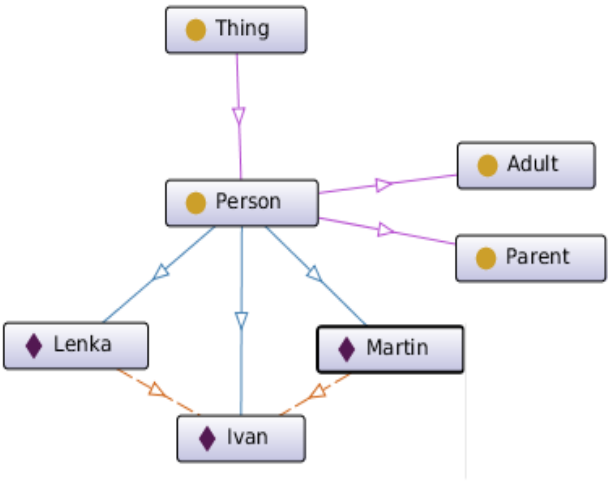

FIGURE 1.3 : EXEMPLE D'UNE REPRÉSENTATION SIMPLIFIÉE D'UNE ONTOLOGIE.

4.1.1. Définitions

En philosophie l'ontologie est une branche de la métaphysique qui s'intéresse à l'étude de l'être en tant qu'être. Elle présente l'étude des propriétés générales de ce qui existe. En informatique une ontologie, selon Tom Gruber [Gruber T, 1993] qu'il définit initialement comme étant une spécification explicite d'une conceptualisation « An explicit specification of a conceptualization », elle permet la spécification dans un langage formel les concepts relatives au domaine avec ces relations. Cette définition a été étendue par Nicola Guarino [Guarino N, 1995] qui a accentué le caractère formel pour assurer l'exploitation et la compréhension des composantes des ontologies par les machines et les systèmes informatiques. Ces systèmes devront être capables d'interpréter la sémantique intégrée dans les informations fournies. Ainsi, une ontologie est une représentation explicite de la sémantique d'un domaine.

Entre temps, les travaux de *Sean B* [Sean B, 2003] ont permets la définition de l'ontologie comme étant un artefact de l'ingénierie constituée et développé autour d'un vocabulaire spécifique. Elle est utilisée pour décrire une certaine réalité avec un ensemble d'hypothèses explicites et de règles sur le sens exprimé des concepts du vocabulaire. Ainsi, une ontologie décrit une spécification formelle d'un certain domaine. C'est est modélisation de connaissances spécifiques partagée et un modèle formel d'inférence exploitable par les systèmes informatique.

Cependant, nous trouvons dans les travaux de *Tom Gruber* [Gruber T, 1993] la précision de la définition de point de vu le contexte de l'informatique et des sciences de l'information. Il a défini que l'ontologie est un ensemble de primitives de représentation pour modéliser un domaine de connaissance. Les primitives de représentation sont généralement des ensembles,

des classes, des attributs, des propriétés) et des relations entre les classes. Les définitions des primitives de représentation incluent des informations cohérentes sur les aspects : significations, contraintes et applications. En pratique, de point de vu structure et modèles, les modèles et langages de description utilisés dans ontologies ont une puissance expressive plus proche de la logique du premier ordre que celle des langages utilisés pour à celles des bases de données. A cet effet, nous trouvons que les ontologies sont considérées comme un niveau « sémantique », tandis que les modèles et les schémas de bases de données sont considérés comme des niveaux « logique » ou « physique ».

Aussi, nous trouvons les travaux de *Fankam C et al* [Fankam C & al, 2009] qui ont caractérisés les ontologies comme une représentation formelle, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine. En effet, les ontologies sont des modèles de représentation des connaissance formelles puisqu'elles permettent des raisonnements automatiques ayant pour objectif d'inférer de nouveaux faits et d'effectuer des vérifications de consistance. Entre autres, elles sont consensuelles c'est à dire admise par l'ensemble des membres et des systèmes d'une communauté. De plus, chaque entité ou relation décrite dans l'ontologie peut être directement référencée par un symbole, à partir de n'importe quel contexte.

En effet, la représentation des connaissances par les ontologies est une alternative très prometteuse pour la modélisation de la sémantique emboité dans le contenu des ressources informationnelles. Ces ontologies permettent de fournir :

- Une structure conceptuelle de base à partir de laquelle il est possible de développer des systèmes à base de connaissances qui soient partageables, et réutilisables.
- L'interopérabilité entre les ressources d'information et de connaissances.

4.1.2. Composantes d'une ontologie

Malgré les différentes définitions sur les ontologies, on trouve qu'il y a un consensus sur ses composants. Parmi les travaux de recherches effectués sur les ontologies et leurs composants nous trouvons les travaux de Gomez Perez [Gomez P, 1999]. Ces travaux confirment que l'exploitation des connaissances traduites par les ontologies s'effectuent à l'aide des composantes suivants : concepts, relations, fonctions, axiomes et instances.

- Les concepts ou les classes : qui présentent l'abstractions pertinentes du domaine. Elles sont retenues en fonction des objectifs et les connaissances pour l'ontologie et ces applications. Ces concepts peuvent être classés selon plusieurs dimensions telles que le niveau d'abstraction : concret ou abstrait, l'atomicité : élémentaire ou composée et le niveau de réalité : réel ou fictif.
- Les relations qui traduisent les associations pertinentes qui existent entre les concepts. Ces relations incluent les associations de généralisation/spécialisation (sous-classe-de), d'agrégation ou de composition (partie-de), …etc. En effet, elles permettent la modélisation de sa structure ainsi que les relations inter et intra concepts

- Les fonctions qui sont des cas particuliers des relations. Dans lesquelles, un élément de la relation est défini en fonction des éléments précédents.
- Les axiomes qui sont des expressions qui sont toujours vraies. Leurs utilisations dans les ontologies permettent plusieurs objectifs comme : la définition de la signification des composants, la définition des restrictions sur les valeurs des attributs, la définition des arguments des relations et la vérification de la validité des informations spécifiées ou en déduire de nouvelles.
- Les instances qui constituent la définition extensionnelle de l'ontologie. Ces objets couvrent les connaissances statiques ou factuelles à propos du domaine du connaissances cible.

4.2. WordNet comme ontologie générale pour l'indexation sémantique

WordNet est une base de données lexicales crée par un groupe de psychologues et de linguistes du laboratoire de sciences cognitives de l'université de Princeton. Initialement, l'objectif de ce projet est la construction d'une ressource lexicale qui permette l'exploitation des mots en tenant compte de leurs relations dans sans contexte conceptuelle ou ontologique. Ainsi, ce réseau lexical de WordNet est modélisé sous la représentation conceptuelle « Lexical Conceptual Graph-LGC » proche du contexte ontologique [Guarino N & al, 1999].

En effet, Le contenu de WordNet, couvre la majorité des noms, verbes, adjectifs et adverbes de la langue Anglaise. Sa dimension ainsi que le domaine de la langue générale qu'il traite lui permettent souvent de couvrir les sujets traités dans les collections de test conventionnelles de la RI comme TREC, CLEF, …etc.

WordNet est un réseau composé de 155 287 nœuds structuré autour de 121 012 mots ou concepts appelés « Synsets ». Le Tableau 1.2 présente une description statistique sur les concepts dans la base de données de WordNet dans sa version 3.1.

Catégorie	Mots	Concepts	Total Paires « Mots-Sens »
Nom	117 798	82 115	199 913
Verbe	11 529	13 767	25 296
Adjectif	21 479	21 509	42 988
Adverbe	4 481	3 621	8 102
Total	**155 287**	**121 012**	**276 299**

TABLEAU 1.2 : DESCRIPTION STATISTIQUE DES CONCEPTS WORDNET (VER. 3.1) [5]

[5] *WordNet* Ver. 3.1 disponible à l'adresse http://wordnetcode.princeton.edu/wn3.1.dict.tar.gz

Dans WordNet, les entrées sont des concepts représentés par des Synsets contenants l'ensemble des termes synonymes qui peuvent désigner un concept. En effet, les concepts sont reliés sémantiquement par des relations affectées aux Synsets et sont représentés par des classes. Les relations de base définie dans WordNet entre les termes est la Synonymie. Entre autres, nous trouvons d'autre relations pour les Synsets comme les relations de type hyponyme-hyperonyme « is-a », et les relations de type meronymie-holonymie, la figure 1.4 présente une illustration simple de ces relations.

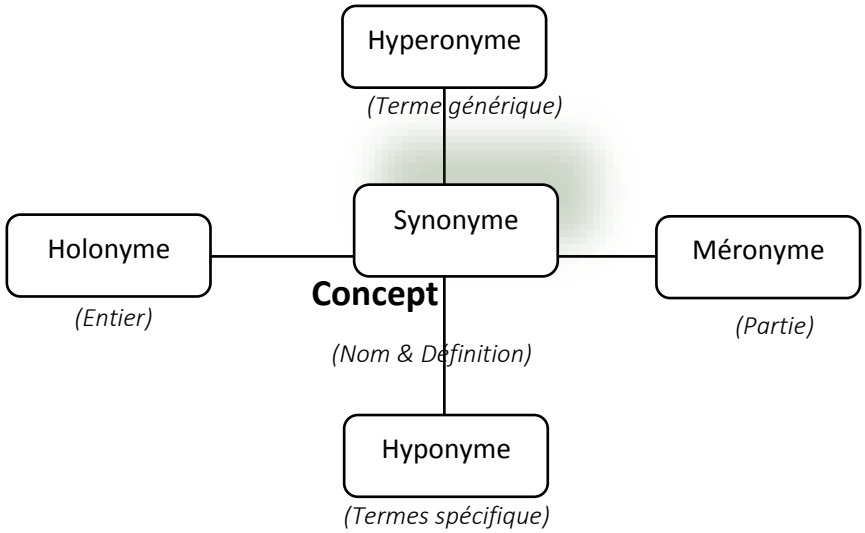

FIGURE 1.4: PRINCIPALES RELATIONS SÉMANTIQUES DANS L'ONTOLOGIE WORDNET.

Cependant, différents types de relations entre les concepts sont définies dans WordNet, parmi eux nous citons :

- La Synonymie : C'est l'association d'un mor à un concept.
- L'hyperonymie : C'est le terme générique utilisé pour désigner une classe englobant des instances de classes plus spécifiques
- L'hyponymie : C'est le terme spécifique utilisé pour désigner un membre d'une classe.
- La Méronymie : Le nom d'une partie constituante, substance ou membre d'une autre classe.
- L'holonymie : Le nom de la classe globale dont les noms Méronymes font partie.

En effet, il existe autres relations moins utilisées comme : la relation « Domain », la relation « Antonymy » qui exprime les sens opposés pour les Synsets, la relation « Troponymy » qui représente la similitude partielle entre les verbes, …etc.

5. Conclusion du chapitre

Dans ce chapitre, nous avons présenté les concepts et techniques liées au domaine d'indexation sémantique pour les ressources textuelle, En effet, nous avons présenté le principe d'indexation ainsi que les ressources sémantiques utilisées comme les ontologies et le WordNet.

Dans ce chapitre nous avons cité les ontologies comme une ressource d'indexation sémantiques, mais nous visons dans ce travail d'utiliser d'autre ressources qui peuvent simuler les taches des ontologies ou WordNet, Pour cela nous détaillons dans le chapitre suivant les techniques de « Word Embedding » comme étant une source pour l'indexation sémantique.

Chapitre 2

Les Techniques de « Word Embedding »

1. Introduction

Le terme « *Deep Learning* » en français « *l'Apprentissage Profond* » a été introduit pour la première fois dans les travaux de *Dechter.R* et *al* [Dechter R & al, 1986] pour le domaine de *« Machine Learning »* et les réseaux neurones artificiels. Ces techniques de « *Deep Learning* » proposées sont basées sur les réseaux de neurones artificiels [Aizenberg I & al, 2013] et peuvent gérer une grande masse de données dans plusieurs étapes de calculs. Elles exploitent des modèles d'apprentissage profond aussi peuvent extraire des attributs à partir des données et des informations brutes à travers différentes couches de prétraitement constituées de divers changements linéaires et non linéaires et recherchent des valeurs bien ordonnées dans chaque couche avec une série de calculs statistiques. À vrai dire, l'apprentissage profond est utilisé comme un support de résolution des problèmes d'intelligence artificielle, des systèmes automatiques tels que « *Word Embedding* » ainsi des techniques de traitement de langage naturel TAL « *Natural Langage Processing* ».En effet, nous trouvons l'utilisation des techniques de « *Deep Learning* » pour apprendre des modèles qui présentent le contexte utilisé pour faire des analyses sur langage naturel tel que les ressources textuelles et même les ressources parlées après une phase de transcription automatique. En d'autres termes, les techniques de « *Deep Learning* » ont pour but d'accroitre la capacité des systèmes informatiques à comprendre et résoudre des problèmes modernes très complexes.

2. L'Apprentissage profond « *Deep Learning* »

2.1. Principes

D'après les travaux de *Deng L* et *al* [Deng L & al, 2014], le « *Deep Learning* » est une classe de techniques d'apprentissage automatique appartient au domaine de « *Machine Learning* » (voir figure 2.1) dans lesquelles de multiples couches de traitement de calcul itératif dans des architectures hiérarchisées supervisées sont exploitées pour les algorithmes d'apprentissage non supervisé pour des tâches d'analyse et de classification. L'apprentissage profond consiste essentiellement à calculer des caractéristiques hiérarchiques des paramètres des réseaux de neurones artificiels pour les représentations vectorielles des données d'observation ou d'entrées. La famille des méthodes d'apprentissage en profondeur s'enrichit de plus en plus, englobant celles des réseaux de neurones, des modèles probabilistes hiérarchiques, ainsi que de nombreux algorithmes d'apprentissage des fonctionnalités supervisés et non supervisés.

Comme tous les réseaux de neurones l'apprentissage profond a besoin à des paramètres d'entrée *X* pour des observations de sorties et résultats *Y.* L'apprentissage automatique est une méthode d'apprentissage statistique dans laquelle chaque instance d'un jeu de données est décrite par un ensemble de caractéristiques ou d'attributs. En revanche, le terme « *Deep Learning* » est une méthode d'apprentissage statistique qui extrait des caractéristiques ou des

attributs à partir de données brutes. Il utilise pour cela des réseaux de neurones comportant de nombreuses couches cachées, des données volumineuses et de puissantes ressources de calcul. Les termes semblent quelque peu interchangeables, cependant, avec les méthodes d'apprentissage profond, l'algorithme construit automatiquement des représentations des données. En revanche, les représentations de données sont codées en dur comme un ensemble de fonctionnalités dans des algorithmes d'apprentissage automatique, nécessitant des processus supplémentaires tels que la sélection et l'extraction des paramètres.

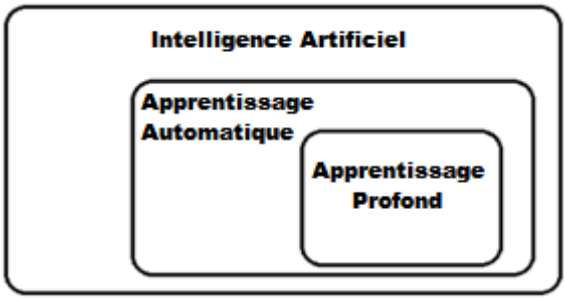

FIGURE 2.1 : LE CONTEXTE DE DEEP LEARNING.

2.2. Les techniques de « *Deep Learning* »

En général, un réseau de neurones artificiels est une technique conçue pour simuler l'activité du cerveau humain et ces neurones biologiques. En particulier, il est utilisé pour les tâches de reconnaissances automatiques des formes et des systèmes informatiques intelligents. L'apprentissage profond est une approche spécifique utilisée pour la construction d'une topologie des réseaux de neurones, pour le traitement des grandes masses des données sur des tâches d'apprentissage automatique, chaque réseaux de neurones fournit des bons résultats pour des types de données spécifiques tels que (vidéo , document , audio …). L'avantage de ses techniques de « *Deep Learning* » se réside dans l'identification manuelle des caractéristiques du données d'une façon automatique dans la partie des couches cachées de processus d'apprentissage. Il facilite ainsi et accélère le temps d'apprentissage du réseau de neurones et permets d'obtenir de meilleurs résultats. Parmi les techniques de « *Deep Learning* » utilisé nous trouvons les réseaux de neurones convolutifs et récurrent défini comme suite :

2.2.1. Réseau de neurones convolutifs (CNN)

D'après les travaux de *Yin W* et *al* [Yin W & al, 2017], les réseaux de neurones convolutif « *CNN* » sont des types spécifiques de réseaux de neurones artificiels qui utilise des perceptrons et un algorithme d'unité d'apprentissage automatique pour l'apprentissage supervisé, permettant d'analyser des données. Les CNN s'appliquent beaucoup plus au traitement d'images, aussi au traitement du langage naturel « NLP » et à d'autres types de tâches cognitives. Comme d'autres types de réseaux de neurones artificiels, un réseau de neurones convolutionnel comporte une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées (voir figure 2.2). Certaines de ces couches sont convolutionnelles

et utilisent un modèle mathématique et algébrique pour transmettre les résultats aux couches successives.

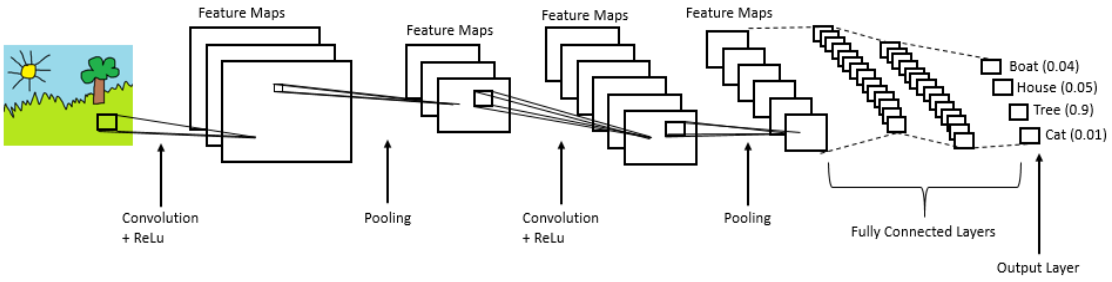

FIGURE 2.2 : L'ARCHITECTURE DES RÉSEAUX DE NEURONES CONVOLUTIFS.[1]

Les réseaux « CNN » sont des exemples typiques de la technique d'apprentissage profond, où un modèle plus sophistiqué accélère l'évolution de l'intelligence artificielle en proposant des systèmes simulant différents types d'activités biologiques du cerveau humain. Généralement les réseaux « CNN » sont composés de quatre principales couches :

- La couche de convolution « *CONV* » qui traite les données d'entrées ou les résultats des couches intermédiaires ou cachées.
- La couche de *Pooling* « *POOL* », qui permet de compresser l'information en réduisant la taille de l'image intermédiaire.
- La couche d'activation, c'est des fonctions de rectification linéaire souvent appelée par « *ReLU* » qui permet la mise à jour des paramètres de la couche actuelle ainsi que les couches précédentes
- La couche entièrement connectée « *Fully Connected* », qui est une couche de type perceptron

2.2.2. Réseau de neurones récurrent « *RNN* »

D'après les travaux de *Yin W* et *al* [Yin W & al, 2017], Les réseaux neurones récurrents *« RNN »* sont des types de réseaux de neurones artificiels avancés « ANN ». Ils intègrent les notions de continuités temporelles dans les entrées de ces couches de réseaux de neurones ces réseaux de neurones récurrents sont adaptés pour des données d'entrée de taille variable (voir figure 2.3). Ils conviennent en particulier pour l'analyse de séries temporelles. Ils sont utilisés en reconnaissance automatique de la parole ou de l'écriture manuscrite - plus en général en reconnaissance de formes - ou encore en traduction automatique.

1 https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

De point de vue dans l'application pratique, les RNN ont été un domaine de concentration actif pour de nombreux professionnels, notamment pour le traitement d'images, le traitement du langage naturel et même pour les modèles qui ajoutent des caractères au texte. Le RNN a été en mesure de générer des résultats textuels démontrant l'aptitude à apprendre l'anglais à partir de rien ou à partir d'entrées de programmation très limitées.

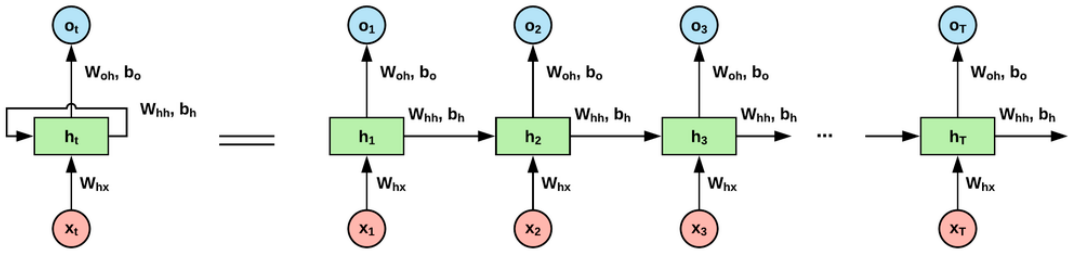

FIGURE 2.3 : L'ARCHITECTURE DES RÉSEAUX DE NEURONES RÉCURRENTS.[2]

De point de vue technique. Les réseaux de neurones récurrents utilisent les mêmes techniques d'apprentissage utilisées dans les réseaux de neurones classiques tels que l'algorithme de rétropropagation du gradient. Cependant, ces réseaux sont limités par le problème de disparition du gradient pour le traitement de grandes séquences pour mémoriser les évènements passés. Nous trouvons dans les littératures des architectures particulières des « *RNN* » qui répondent à ce problème tel que les réseaux « *LSTM* » « *Long Short-Term Memory* » et « *Word Embedding* ». Ces techniques permettent d'étudier les comportements des réseaux de neurones récurrents avec la théorie des bifurcations, mais la complexité de cette architecture augmente très rapidement avec le nombre de neurones.

2.3. Les outils de « *Deep Learning* »

Dans la pratique, nous trouvons plusieurs plateformes, bibliothèques et environnements de programmation qui ont était développés pour les différents techniques et modèles de l'architecture des réseaux récurrentiel et apprentissage profond « *Deep Learning* » dont les langages de développements les plus sollicités sont : *Java*, *MATLAB, c/c++, CUDA* et *python* ...etc.

2.3.1. La bibliothèque Gensim

D'après le site officiel de *Gensim*[3], nous trouvons que cette bibliothèque est gratuite en *Python* et possède une sémantique statistique évolutive. Elle permet l'analyse les documents textuels bruts ainsi que la recherche des structures sémantiques et la récupération des informations sémantiquement similaires. La bibliothèque « *Gensim* » est sous licence GNU

2 http://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/vanilla-rnn-for-classification
3 https://radimrehurek.com/gensim/

LGPLv2.1 approuvée par l'OSI. Cela signifie qu'il est gratuit pour un usage personnel et commercial, mais que si les utilisateurs apportent des modifications dans les fonctions de la bibliothèque « *Gensim* » qu'ils distribuent à d'autres personnes, ils doivent divulguer le code source de ces modifications. En dehors de cela, les utilisateurs sont libres de redistribuer la bibliothèque « *Gensim* ». Cette bibliothèque est caractérisée par :

- La capacité de sa sémantique statistique évolutive
- L'analyse des documents en texte brut pour la structure sémantique
- La récupération des informations et concepts sémantiques similaires

En plus, la bibliothèque « *Gensim* » peut traiter de grands corpus à l'échelle du Web en utilisant des algorithmes d'apprentissage séquentiel récurrent. Il n'est pas nécessaire que l'ensemble du corpus d'entrée réside entièrement dans la RAM à un moment donné. De plus, les principaux algorithmes du Gensim utilisent des routines mathématiques hautement optimisées. Aussi, la bibliothèque « *Gensim* » contient également une version distribuée de plusieurs algorithmes, destinés à accélérer le traitement et la récupération sur des clusters de machines. Cependant, pour le langage *Python*, la bibliothèque « *Gensim* » fonctionne sous les systèmes d'exploitation : *Linux*, *Windows* et *OS X*, ainsi que sur toute autre plate-forme prenant en charge *Python* et la bibliothèque de calculs matriciels « *NumPy* ». En plus, la bibliothèque « *Gensim* » contient aussi des implémentations fiables pour les variantes des réseaux de neurones récurent tel que *SVMlight* et *LSTM*.

2.3.2. La bibliothèque TensorFlow

« *TensorFlow*[4]» est une bibliothèque de logiciels open source pour le calcul numérique à l'aide des techniques du domaine de la théorie de graphes et les structures d'arbres binaires. Elle exploite les concepts des nœuds d'arbres pour la représentent graphique des opérations mathématiques, tandis qu'elle utilise les concepts d'arcs orientés pour la représentation des données sous forme des tableaux de données multidimensionnels (tenseurs). L'architecture des structures d'arbres dans la théorie des graphes permette aux utilisateurs de déployer des calculs sur un ou plusieurs processeurs ou calculs graphique via des plateformes « GPU » sur un ordinateur de bureau ou sur des serveurs locaux ou même sur le cloud via les « *API TensorFlow*[5] » et elle contient plusieurs couches dans son architecture de Framework comme la présente la figure 2.4. La bibliothèque « *TensorFlow* » a été développée à l'origine par des chercheurs et des ingénieurs travaillant au sein de *« Google Brain Team »* au sein de l'organisation de recherche Machine Intelligence de Google dans le but de mener des recherches sur le domaine de « *Machine Learning* » et l'apprentissage profond, mais le système est suffisamment général pour être applicable dans de nombreux autres domaines,

4 https://www.tensorflow.org
5 http://www.deeplearning.net/software/tensorflow/

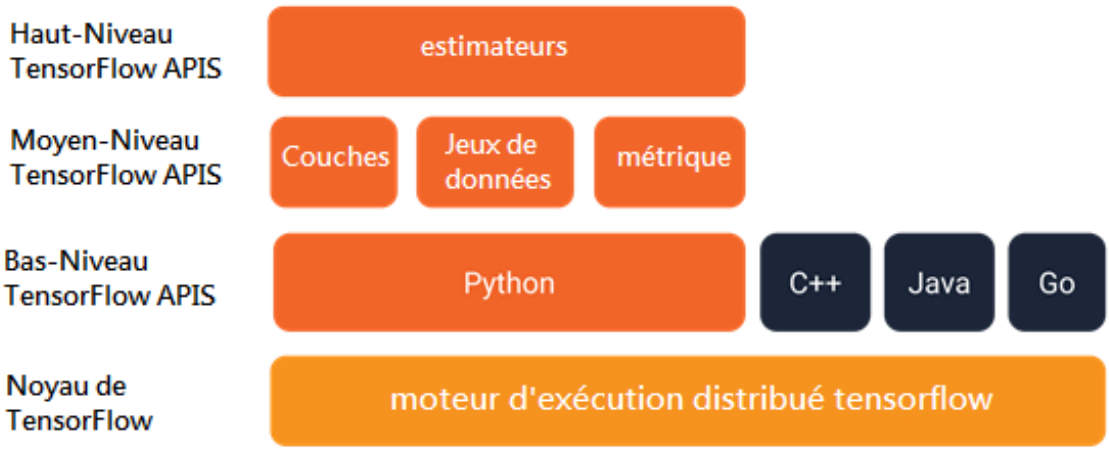

FIGURE 2.4 : ARCHITECTURE DE LA BIBLIOTHÈQUE TENSORFLOW.[6]

La bibliothèque « *TensorFlow* » fournira alors les outils nécessaires pour assembler les sous-graphes communs aux réseaux de neurones, mais les utilisateurs peuvent écrire leurs propres bibliothèques de niveau supérieur sur « *TensorFlow* ». En pratique, nous trouvons que la définition de nouvelles compositions d'opérateurs pratiques est aussi simple que d'écrire une fonction *Python*. L'utilisation de la bibliothèque « *TensorFlow* » permet aux chercheurs et développeurs d'intégrer plus rapidement des idées à des produits et de partager du code plus directement et avec une reproductibilité scientifique accrue. Les utilisateurs de « *TensorFlow* » définissent l'architecture de calcul de leur modèle prédictif, via les classes et méthodes disponibles pour tous les composants de l'architecture de réseaux neurones ave l'apprentissage profond « *Deep Learning* ». Entre autres, « *TensorFlow* » est disponible avec une interface *Python* facile à utiliser et une interface C ++ pratique pour construire et exécuter des graphes de calcul. L'écriture des programmes TensorFlow Python ou C ++ [McClure N, 2017] et [Abrahams & al, 2016].

2.3.3. La bibliothèque Keras

D'après le site officiel de « *Keras*[7] », il est une bibliothèque d'apprentissage profond pour la bibliothèque « *Theano* » (elle sera détaillée dans la section suivante) et la bibliothèque « *TensorFlow* ». Il s'agit d'une bibliothèque de réseaux de neurones embarqué, écrit en *Python* et capable de s'exécuter sur « *TensorFlow* » ou « *Theano* ». Elle a été développée pour permettre une expérimentation rapide comme la présente la figure 2.5. La bibliothèque d'apprentissage profond de « *Keras* » permet un prototypage simple et rapide. Elle prend en charge les réseaux convolutionels et les réseaux récurrents, ainsi que leurs combinaisons. La bibliothèque « *Keras* » prend également en charge des schémas de connectivité arbitraire et s'exécute de manière transparente sur le processeur et le processeur graphique. Les structures

6 https://www.tensorflow.org/guide/premade_estimators
7 https://keras.io/

de données de base utilisées dans la bibliothèque « *Keras* » sont les modèles séquentiels, les piles linéaires et les arbres binaires. Les principes directeurs de Keras incluent la modularité. En particulier, les couches neuronales, les fonctions de coût, les optimiseurs, les schémas d'initialisation, les fonctions d'activation, les schémas de régularisation sont tous des modules autonomes que les utilisateurs peuvent combiner pour créer de nouveaux modèles. De point de vue programmation, et nouveaux modules sont simples à ajouter (en tant que nouvelles classes et fonctions), et les modules existants fournissent de nombreux exemples. Le fait de pouvoir créer facilement de nouveaux modules permet une expressivité totale, ce qui permet d'utiliser la bibliothèque de « *Keras* » pour des recherches avancées [Moolayil J & al, 2019].

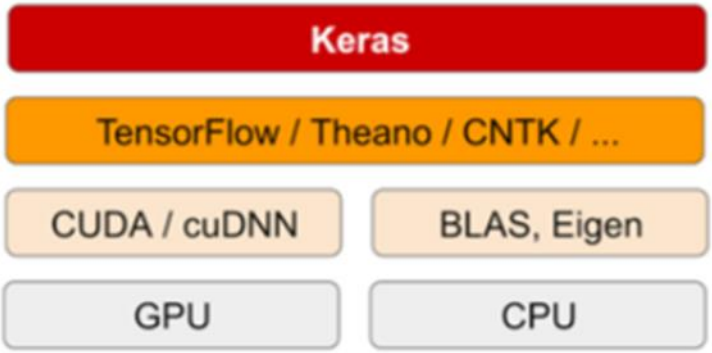

FIGURE 2.5 : ARCHITECTURE DE LA BIBLIOTHÈQUE [8]

2.3.4. La bibliothèque Theano

D'après le site officiel « *Theano* [9] » c'est une bibliothèque *Python* qui permet aux utilisateurs de définir, d'optimiser et d'évaluer des expressions mathématiques, en particulier celles contenant des tableaux multidimensionnels. Elle permet d'accélérer les calculs et atteindre des vitesses équivalentes aux mises en œuvre par le langage C conçues à la main pour les problèmes impliquant de grandes quantités de données utilisant « *Theano* ». Elle peut également dépasser les routines de programmation en langage C en exploitant les accélérateurs graphiques récents dans un mode de calculs GPU. « *Theano* » combine les aspects d'un système de calcul algébrique *« CAS »* et d'un compilateur optimiseur. Il peut également générer du code *C* personnalisé pour de nombreuses opérations mathématiques. Cette combinaison de *« CAS »* avec une compilation d'optimisation est particulièrement utile pour les tâches dans lesquelles des expressions mathématiques complexes sont évaluées de manière répétée et pour lesquelles la vitesse d'évaluation est cruciale. Dans les situations où de nombreuses expressions différentes sont évaluées une seule fois, la bibliothèque « *Theano* » peut minimiser le temps système nécessaire à la compilation et à l'analyse, tout en offrant des fonctionnalités symboliques telles que la différenciation automatique. Le compilateur de « *Theano* » applique de nombreuses optimisations de complexité variable à ces expressions symboliques. « *Theano* » n'est pas un langage de programmation au sens

8 https://towardsdatascience.com/my-journey-into-deeplearning-using-keras-part-1-67cbb50f65e6
9 www.deeplearning.net/software/theano/

habituel du fait que les utilisateurs écrivent un programme en *Python* qui construit des expressions pour « Theano ». Cette bibliothèque se concentre davantage sur les expressions tensorielles que les matrices multidimensionnelles « *Numpy* » et dispose de plus de machines pour la compilation. Si « *Numpy* » doit être comparé à *MATLAB* et à *Mathematica*, Theano est une sorte d'hybride des deux qui tente de combiner le meilleur des deux modules [Bourez C, 2017].

2.3.5. La bibliothèque Scikit-learn

D'après le site officiel de « *Scikit-learn*[10] » est une bibliothèque « Machine Learning » d'apprentissage automatique open source pour le langage de programmation *Python*. Elle comporte divers algorithmes de classification, de régression et de clustering, notamment des de supports vecteurs machines SVM, les arbres de décision, la descente du gradient, k-means, et elle est conçue pour interopérer avec le langage *Python* et ces bibliothèques numériques de calculs scientifiques tels que « *NumPy* » et *« SciPy »*. La bibliothèque « *Scikit-learn* » intègre une implémentation de plusieurs algorithmes tels que :

a) *Les algorithmes de classification* : elles permettent d'identifier à quelle catégorie ou classe un objet il appartient

- *Applications* : Détection de spam, Reconnaissance d'image. *Algorithmes* : SVM, voisins les plus proches, l'arbre de décision.

b) *Les algorithmes de régression* : elles prédirent les attributs à valeur continue associée à un objet.

- *Applications :* Réponse aux médicaments, Prix des actions. Algorithmes : SVR, régression, assurances.

c) *Les algorithmes de segmentation.* Elles permettent le regroupement automatique d'objets similaires dans des ensembles.

- *Applications :* segmentation de la clientèle, résultats des expériences de regroupement. Algorithmes : k-means, classification spectrale, décalage moyen.

2.3.6. La bibliothèque Deeplearning4j

D'après le site officiel de « *Deeplearning4j* [11] » est la première bibliothèque d'apprentissage profond distribué et open source de qualité commerciale écrite pour Java et Scala. Intégré à Hadoop et Spark. « *Deeplearning4j »* est conçu pour être utilisé dans les environnements professionnels utilisant des processeurs graphiques et des processeurs distribués. Le Framework « *Skymind* » est son support commercial. « *Deeplearning4j »* a pour objectif d'être un plug-and-play de pointe, plus conventionnel que configuration, ce qui permet un prototypage rapide pour les non-spécialistes de programmation. « *Deeplearning4j »* est personnalisable à l'échelle. Publiés sous la licence Apache 2.0, tous

10 https://scikit-learn.org/
11 https://deeplearning4j.org/

les dérivés de « *Deeplearning4j* » appartiennent à leurs auteurs. « *Deeplearning4j* » peut importer des modèles de réseaux neuronaux à partir de la plupart des infrastructures majeures via « *Keras* », y compris « *TensorFlow* » et « *Theano* ». Entre autres, « *Deeplearning4j* » permet aux utilisateurs de composer des réseaux de neurones profonds à partir de divers réseaux peu profonds, chacun d'entre eux formant une « couche ». Cette flexibilité permet aux utilisateurs de combiner des machines « Boltzmann » restreintes, d'autres auto-encodeurs, des réseaux convolutifs ou récurrents, selon les besoins, dans une infrastructure distribuée de niveau production qui fonctionne avec *Spark* et *Hadoop* sur des processeurs ou des *GPU* distribués.

En pratique, il existe de nombreux paramètres à ajuster lorsque les utilisateurs forment un réseau d'apprentissage profond. Le « *Deeplearning4j* » peut servir d'outil simple aux programmeurs *Java*, *Scala*, etc. En effet, la bibliothèque « *Deeplearning4j* » pour ces récentes versions distribuées, notamment *Hadoop*, *Apache* et *Spark* est utilisée pour accélérer l'apprentissage. Les bibliothèques sont complètement open source, Apache 2.0, et maintenues par la communauté des développeurs et par l'équipe « *Skymind* ». la bibliothèque « *Deeplearning4j* » est écrite en Java et est compatible avec tous les langages JVM, tels que *Scala*, *Clojure* ou *Kotlin*. Les calculs sous-jacents sont écrits en *C*, *C ++* et *Cuda.*

2.3.7. Les plateformes Cuda « Cuda-convnet2 »

D'après son site officiel, on trouve que « *Cuda*[12] » ou « *Cuda-convnet2* » est une branche de « *Nervana* [13] » dans « *Alex Krizhevsky* [14] », « *Cuda-convnet2* » contient plusieurs extensions, notamment : un nouveau backend en python, appelé « *CudaNet* », destiné à être intégré dans le Framework néon de « *Nervana* ». Ils ont également inclus plusieurs nouveaux noyaux et fonctions prenant en charge des coûts tels que les coûts multivoies, l'interface python vers la mémoire du GPU, le support des noyaux sans texture, les comparaisons matricielles ou scalaires, ainsi que la normalisation du contraste local. Cette version propose également une installation basée sur une ligne de commande « *pip* » ou « *cmake* » ainsi que des vérifications et des corrections supplémentaires. Pour l'installation de la structure, les utilisateurs doivent s'assurer qu'ils ont satisfait à tous les packages de dépendance requis, y compris l'installation de la boîte à outils *« CUDA* » et du kit de développement logiciel « *SDK CUDA* ». Cette plateforme ou bibliothèque permet l'exploitation des cartes d'accélération graphique de type GPU. Cela inclut les puces « *GK110* » et « *GK114* », qui peuvent être trouvées sur les *GPU Tesla K20, Tesla K20x, Tesla K40, GForce Titan et GForce GTX 780*. Cependant, pour les GPU plus anciens, y compris les GPU basés sur GK104 tels que Tesla K10 et GeForce 680, ne fonctionneront pas.

Le projet initial de « *Cuda-convnet2* » comportait trois nouvelles fonctionnalités principales par rapport à « *Cuda-convnet* ». Le premier était l'amélioration des temps

12 https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html
13 https://github.com/NervanaSystems/
14 https://github.com/akrizhevsky/

d'apprentissage sur les *GPU Nvidia* de génération *Kepler (Gforce Titan, K20, K40)*. Il disposait également d'un support d'apprentissage multi-*GPU* mettant en œuvre le parallélisme des données, le parallélisme de modèle et les approches hybrides pour la mise en parallèle des réseaux de neurones de convolution. [Michael F & al, 2015].

3. La technique de « Word Embedding »

3.1. Introduction

La technique de « *Word Embedding* » qui désigne la représentation distributionnelle vectorielle des mots dans les documents joue un rôle important dans la plupart des approches modernes du traitement du langage naturel « *Natural Langage Processing – NLP* ». Ces techniques reposent sur des méthodes pour représenter la distribution des mots avec une représentation vectorielle de chaque mot avec un vecteur de taille fixe selon leurs contextes. Les techniques de « Word Embedding » offrent la possibilité de création des modèles qui représentent les documents et ces termes avec un modèle vectoriel de taille fixe qui prend en charge les relations sémantiques et syntaxiques entre ces termes et ressource.

3.2. Principes

Dans les littératures nous trouvons la définition du concept « *Word Embedding* » comme une représentation numérique et vectorielle d'un mot dans un espace multidimensionnel selon [Dima S & Arafat S, 2018]. Dans ce contexte, chaque dimension représente une caractéristique latente pour décrire le sens de mot. Ainsi, ils utilisent des méthodes algébriques et statistiques pour détecter les contextes relatifs aux termes par rapport aux leurs propriétés lexicale, sémantique et syntaxique selon un corpus d'entrée. En effet, nous trouvons plusieurs techniques automatiques de capture de contexte, parmi ces techniques utilisées nous trouvons : *word2vec*, *fastText*, *Glove*. Les techniques « *Word Embedding* » sont utilisées pour la résolution de plusieurs problèmes dans le domaine de recherche d'informations comme les techniques d'indexation et les systèmes de traitement automatiques du langage naturel [Gysel R & al, 2018] et [Azpiazu M & al, 2018], [Santos D.C & al, 2014] et [Young & al, 2018].

3.3. Les implémentations de « Word Embedding »

3.3.1. Le modèle Word2vec

D'après les travaux cités dans [Mikolov T & al, 2013a], nous trouvons l'implémentation de la technique de « *Word Embedding* » avec l'algorithme « *Word2Vec* ». Cette implémentation est un outil open source basé sur un apprentissage profond. L'algorithme « Word2Vec » est un modèle prédictif efficace en termes de calcul pour l'apprentissage de l'intégration de mots à partir de texte brut et non structuré [Mikolov T & al, 2013b]. Dans l'algorithme « *Word2vec* » nous trouvons la représentation des mots avec des vecteurs de taille fixe. Ces vecteurs sont capables de capturer efficacement les similarités sémantiques et syntaxiques existantes entre ces termes selon le corpus d'apprentissage utilisé.

Par exemple, nous trouvons que les concepts « *Queen* » et « *Women* » ainsi que « man » « *King* » sont sémantiquement similaires. Le « *Word2vec* » est une approche basée sur la prédiction par les techniques d'apprentissage à partir des ressources textuelles. Le « *Word2vec* » utilise une architecture basée sur l'apprentissage profond avec les réseaux de neurones artificiels. En effet, c'est le modèle de « *Word Embedding* » le plus populaire actuellement. En effet, nous trouvons cette implémentation comme des ressources fondamentales dans les modèles d'apprentissage profond pour le domaine du « *NLP* ». Techniquement, il utilise des modèles de réseau neuronal pour apprendre automatiquement l'apparence de mots dans le corpus et les mapper dans un espace dense de faible dimension. Pour le développement de « *Word2Vec* », l'équipe de *« Google TM »* a utilisé un corpus de 100 milliards de mots non catalogués plus de phrases multi domaines. Le modèle original est disponible sur le Web[15]. Des définitions techniques sur la façon dont il traite le problème de « Word Embedding » peuvent être trouvées dans plusieurs travaux de recherches tels que les travaux de *Mikolov* [Mikolov T & al, 2013b]. Techniquement, nous trouvons que cet algorithme est basé sur deux principales approches :

3.3.1.1. L'approche CBOW

L'approche *« CBOW »* qui est l'abréviation de *« Continuous Bag-Of-Words »* est une approche qui utilise un classifieur non linéaire pour classifier le mot central prédit en fonction des mots adjacents dans le corpus d'apprentissage. L'architecture de *« CBOW »* peut être illustrée à la figure 2.6. La technique « CBOW » permet de maximiser l'équation suivante [Mahdaouy A & al, 2016] :

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\log[p(wt|wt-c,\dots,wt-1,wt+1,\dots,wt+c)] \qquad (01)$$

15 https://github.com/dav/word2vec.

Avec *w(t)* représente le mot actuel, tandis que les mots du contexte sont représentés à l'aide des symboles suivants :

{w(t-c) , ... ,w(t-2) ,w(t-1) , w(t+1) , w(t+2) , ... , w(t+c) }

Et *c* représente la taille de la fenêtre glissante détermine le nombre de mots dans le contexte, par exemple, si la taille de la fenêtre glissante est de cinq, le nombre de mots de contexte est de quatre et la valeur de c sera égale à quatre. De plus, pour prédire un mot, les deux mots précédents et les deux mots suivants, du mot du milieu à prédire, doivent être pris en compte dans le contexte.

3.3.1.2. L'approche Skip-Gram

Les approches « *CBOW* » et « *Skip-gram* » du modèle word2vec sont très similaires dans la structure. Cependant, il n'y a qu'une seule différence entre les deux approches. Tandis que dans « *CBOW* », l'entrée du réseau neuronal correspond aux mots du contexte et la sortie au mot du milieu, dans le modèle « *Skip-gram* », l'entrée est le mot actuel ; le mot du milieu ; et la sortie aux mots du contexte. Par exemple, si la taille de la fenêtre est 5, dans le cas de « CBOW », l'entrée sera constituée de 2 mots de l'historique et de 2 mots futurs, tandis que la sortie sera le mot du milieu. Dans le cas de « *Skip-gram* », l'inverse est vrai, l'entrée sera le mot du milieu et la sortie contiendra 2 mots futurs et 2 mots de l'historique. De plus, si le nombre de mots de contexte ou la taille de la fenêtre augmente, la qualité du modèle augmente. D'autre part, la complexité de calcul augmentera. L'architecture du modèle « *Skip-gram* » peut être illustrée à la figure 2.6, Les mots de contexte sont représentés à l'aide des symboles suivants :

{w(t-c) , ... ,w(t-2) , w(t-1) , w(t+1) ,w(t+2) , ... , w(t+c) }

Tandis que *w (t)* représente le mot actuel. Comme dans CBOW, l'objectif du modèle est de maximiser le logarithme de la probabilité dans l'équation (2) [Mahdaouy A & al, 2016].

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{J=t-c,j<>t}^{t+c}\log[p(wj|wt)] \qquad (02)$$

De plus, la fenêtre glissante permet de prédire le mot suivant. Enfin, dans les deux modèles, l'entrée pour le réseau de neurones sera la représentation instantanée des mots [Mikolov T & al, 2013a].

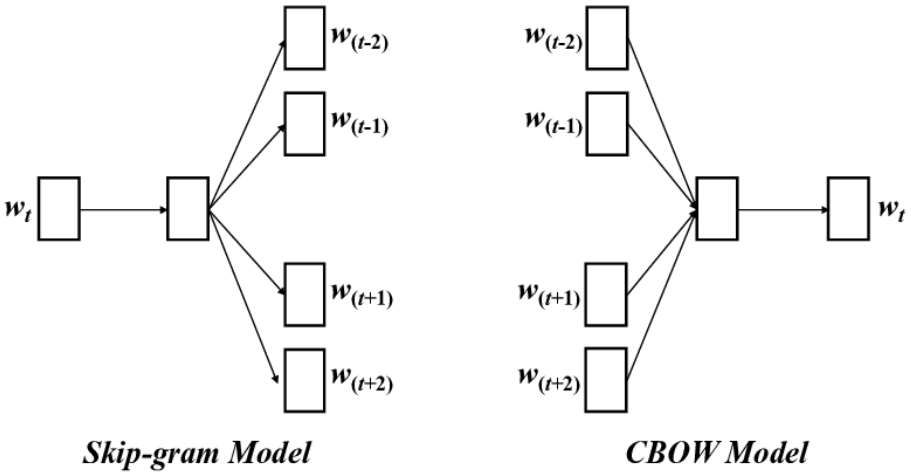

FIGURE 2.6 : LE MODÈLE CBOW ET LE MODÈLE SKIP-GRAM. [MIKOLOV T & AL, 2013A].

3.3.2. Le modèle Glove

Après la publication de l'algorithme « Word2Vec » dans les travaux de *Mikolov* [Mikolov T & al, 2013a], plusieurs travaux de recherches ont utilisé ce modèle dans différents projets et systèmes informatiques. Parmi eux, nous citons les travaux de *Pennington* qui ont introduit l'utilisation du modèle « *Glove* » pour expliquer pourquoi de telles représentations sont efficaces et modifient les processus d'optimisation utilisés pour « Word2Vec » en tant que type spécial de factorisation pour les matrices jointes d'occurrence de mots [Pennington J, 2014. En effet, le modèle *« Glove »* permet d'apprendre les représentations vectorielles des mots en examinant les occurrences de mots dans le corpus. Avant de former le modèle, il est nécessaire de construire une matrice d'occurrence conjointe M, dans laquelle l'élément M_{ij} représente la fréquence du mot i qui apparaît dans le même contexte que le mot j. La méthode ne parcourt le corpus qu'une seule fois pour construire la matrice M. Le modèle utilise uniquement la matrice d'occurrence conjointe pour formuler des imbrications de mots.

3.3.3. Le modèle FastText

Récemment, *« Facebook Research »* a ouvert un projet d'intégration de mots : « *FastText* ». Ce projet a pour objectif de concevoir une méthode rapide et efficace pour apprendre les représentations vectorielles de mots et effectuer les tâches de classification sur les ressources textuelles. L'objectif principal de « *FastText* » est de prendre en compte la structure interne des mots au lieu d'apprendre les représentations des mots. Ceci est remarquablement utile pour les langues riches sur le plan morphologique sachant que les représentations de différentes formes morphologiques de mots seraient apprises de manière indépendante [Pennington J, 2014]. Le modèle « FastText » fonctionne en faisant glisser une fenêtre sur le texte cible et en apprenant le mot central à partir du contexte restant selon les « *CBOW* » ou *« Skip-Gram »*. L'apprentissage automatique dans ce modèle peut être considéré comme une série de mises à jour d'un réseau de neurones à deux couches de poids et trois couches de neurones, les deux couches externes ayant chacune un neurone pour chaque mot du

vocabulaire, et la couche intermédiaire ayant autant de neurones pour la dimension de l'espace d'intégration. Cette approche est très similaire à « *Word2Vec* ». Cependant, contrairement à « *Word2Vec* », le modèle « *FastText* » peut également apprendre des vecteurs pour des sous-parties de mots, appelés *n-grammes* de caractères. Cela garantit, par exemple, que les mots *love*, *loved* et *beloved* ont toutes une représentation vectorielle similaire, même s'ils ont tendance à apparaître dans des contextes différents.

4. Les travaux connexes

Dans cette section nous présentations quelques travaux qui ont utilisé les techniques de « *Word Embedding* » pour des travaux d'indexation et représentation des ressources textuelle. Nous avons essayé de les représenter par rapport à leurs objectifs ainsi que les techniques utilisées.

4.1. « Word Embedding » pour la catégorisation de documents par l'analyse sémantique

La catégorisation de documents consiste à classer automatiquement les documents par rapport aux topiques ou classes préalablement définies. Le principal problème de cette tâche est la technique des représentations utilisée pour modéliser le contenu des documents ; termes ; dans l'espace vectoriel pour puisse exploiter les classifieur statistiques existant dans le domaine de « *Deep Learning* ». D'après [Ju R & al ,2015] les auteurs ont proposé un nouveau modèle appelé « *Latent Semantic Analysis – LSA* » avec l'utilisation du modèle « *Word2Vec* » pour une tache de classification pour des documents textuels. Il s'agit de la première tentative de combinaison de la technique « *Word2Vec* » avec « *Latent Semantic Analysis* » lors de la catégorisation de document. Il permet de mapper un document sur un espace vectoriel en conservant intégralement le contenu de celui-ci. Ils ont créé une matrice terme par document dont l'élément est déterminé par pondération terme-fréquence de document inverse via la métrique « *TF-IDF* » et par le vecteur mot formé par « *Word2Vec* ». Cette matrice est une matrice tridimensionnelle et elle est utilisée pour modéliser le plus possible la signification de chaque mot et le contenu de chaque document. Ainsi, ils ont utilisé la décomposition en valeurs singulières « *Single Values Data -SVD* » sur les valeurs des matrices afin de minimiser la complexité des calculs. Ensuite, les vecteurs de document obtenus à partir du nouveau modèle sont utilisés comme une couche d'entrée pour un classifieur « *CNN* ». Les performances du modèle proposé ont été évaluées sur la base d'un corpus à base de 20 newsgroups. Les résultats montrent que ce nouveau modèle produit de meilleurs effets sur les tâches de catégorisation de documents et une précision améliorée d'environ 15% par rapport aux méthodes traditionnelles, telles que « *LSA* » [Salton G & al,1975] et les « *SVM Support Vector Machine* » [Dumais S.T, 2004].

4.2. « Word Embedding » pour la désambiguïsation non supervisée des acronymes

D'après les travaux publiés dans [Jean C & al, 2018] les chercheurs ont soulevé le problème des abréviations et des acronymes. En effet, les articles scientifiques de toutes les disciplines contiennent de nombreuses abréviations et acronymes. Dans de nombreux cas, ces acronymes sont ambigus. Ils présentent une méthode pour choisir la définition contextuelle correcte d'un acronyme qui ne nécessite pas de formation pour chaque acronyme et peut donc s'appliquer à un grand nombre d'acronymes différents avec seulement quelques cas. Ils ont construit un ensemble de 19954 exemples de 4365 acronymes ambigus extraits de légendes d'images dans des journaux scientifiques, ainsi que leur définition contextuellement correcte dans différents domaines. Ainsi, ils utilisent un modèle *« Word Embedding* » pour tous les mots du corpus et comparons le vecteur de contexte moyen des mots dans l'expansion d'un acronyme avec le vecteur moyen pondéré des mots dans le contexte de l'acronyme. Ils montrent que cette méthode surpasse clairement la similarité en cosinus utilisés dans le domaine de recherche d'information. De plus, ils montrent que le modèle *« Word Embedding* » défini à partir d'un corpus de 1 milliard de mots de textes scientifiques exécute des mots intégrés à partir de corpus généraux beaucoup plus vastes. Ils ont utilisé le texte intégral des 672157 articles de la base de données, soit un total de 1,662 × 109 termes *« Tokens* », pour calculer l'incorporation de mots pour tous les mots dans les définitions des acronymes et des légendes des images. Pour la phase d'apprentissage du modèle vectoriel des mots, ils utilisent uniquement les corps du texte sans les légendes correspondantes. Ils ont été obtenus des meilleurs résultats avec l'utilisation du modèle « *Word2Vec* » avec une dimension de 300 valeurs pour chaqu'un avec « *CBOW =5* » et ceci après une phase de prétraitements textuelle avec le module « *NLTK* ». Pour la validation de leurs résultats, ils ont comparé les résultats obtenus par rapport « *GoogleNews* » [Mikolov & al, 2013a] et « *GloVe »* [Pennington J & al, 2014].

4.3. « Word Embedding » comme un support de similarités

La masse importante des ressources textuelles sur le web engendre des problèmes de recherches et d'accès pour les systèmes de recherche d'informations dans le Web. Dans ce contexte, nous trouvons plusieurs travaux qui proposent des techniques pour faciliter l'exploitation de ces ressources. Parmi eux, nous intéressons à celles qui utilisent les modèles de *« Word Embedding* **»** comme un support de modélisation d'indexation et de recherche.

4.3.1. La distance entre les documents

En effet, il n'existe pas assez de travaux de recherche qui s'intéressent à calculer les distances de document en fonction de mots, en utilisons le modèle *« Word Embedding* **»**. Parmi ces quelques travaux, nous citons les travaux « *Doc2Vec »* qui présente une extension de *« Word2Vec* » pour apprendre les corrélations entre mots et documents qui intègrent des

documents dans le même espace vectoriel où les mots sont incorporés. Ils utilisent l'indexation sémantique latente traditionnelle qui fonctionne directement sur la matrice de termes-documents et génère une représentation vectorielle des documents. [Qv Le & Mikolov T, 2014]. D'autres recherches calculent les distances entre les documents et les mots incorporés sans les incorporer eux-mêmes, comme en utilisant la distance *« Word Mover's »* [Matt J.K & al, 2015]. Aussi, nous trouvons une approche simpliste consistant à prendre la moyenne pondérée des mots incorporés pour représenter des documents et à obtenir des résultats relativement satisfaisants dans un défi de délimitation des topiques [Koopman R & al, 2015] et [Koopman R & al, 2017]. Ainsi, nous trouvons une autre variante qui s'appelle *« Paragraph Vectors »* a récemment été proposé comme méthode non supervisée pour l'apprentissage de représentations distribuées pour des segments de texte, pour la capture de similarités sémantiques de documents dans des représentations vectorielles et leur utilisation pour la classification de critiques de films ou la récupération de pages Web . [Qv Le & Mikolov T, 2014].

4.3.2. Le modèle BOSWE

En vision par ordinateur, le modèle « *BOVW* » **« Bag Of Vector Word »** peut être appliqué à la classification d'images et aux tâches associées, en traitant les descripteurs d'images comme des mots. Un sac de mots visuels est un vecteur de comptage d'occurrences d'un vocabulaire de caractéristiques d'image locales. Le vocabulaire est généralement obtenu par une quantification vectorielle des caractéristiques de l'image en mots visuels. Inspirés du modèle « *BOVW* », les chercheurs proposent une méthode similaire pour traiter les documents texte en utilisant le modèle « Word Embedding ». Dans cette approche conçue pour le texte, les descripteurs d'image sont remplacés par des représentation « Word Embedding ». Sachant que l'incorporation de mots contient des informations sémantiques en projetant des mots sémantiquement semblables dans la même région de l'espace de contexte, ils proposent de regrouper les vecteurs de mots afin d'obtenir des groupes de mots sémantiques pertinents. Chaque centroïde des groupes nouvellement formés peut être considéré comme un super mot. En réunissant les vecteurs super-mots, on obtenir un vocabulaire qu'est utiliser par la suite pour décrire chaque document sous la forme d'un histogramme d'incorporation de super-mots. Ils appellent ce modèle de « Bag Of Super Word Embedding » (BOSWE). [Andrei M.B & al, 2017]. L'architecture de ce modèle est décrite dans la figure 2.7.

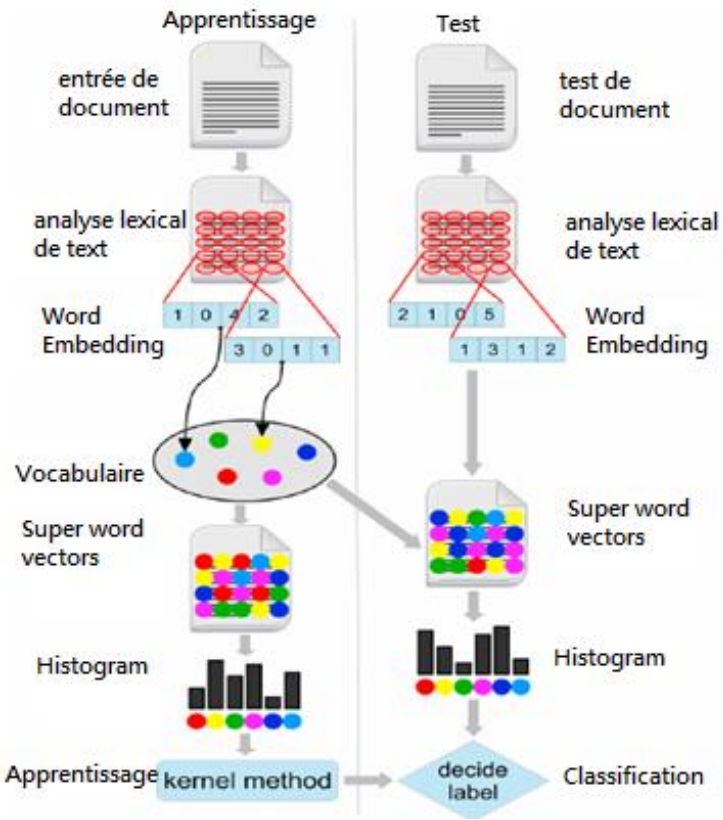

FIGURE 2.7 : LE MODÈLE BOSWE POUR LA CLASSIFICATION DE TEXTE. [ANDREI M.B & AL, 2017]

4.4. « Word Embedding » pour la synthèse de documents

D'après les travaux présentés dans [Kamal A & al, 2018]. Ils présentent la problématique des techniques de résumé automatique des documents pour la masse gigantesque des ressources éparpillées dans le Web. Les auteurs proposent d'utiliser les approches « *LSA* », « *Word Embedding* » et les techniques d'apprentissage profond de « *Deep Learning* ». Ainsi, pour améliorer les schémas de pondération permettant de calculer la matrice d'entrée de l'analyse sémantique latente, deux schémas de pondération basés sur l'intégration sont proposés puis combinés pour calculer les valeurs de cette matrice. Ce sont des versions modifiées du poids augmenté et de la fréquence d'entropie qui combine la force des schémas de pondération traditionnels utilisée « *Word Embedding* ». Cette approche proposée est évaluée sur trois jeux de données anglais, DUC 2002, CIC 2004 et la synthèse multilingue de documents uniques 2015. Les résultats expérimentaux sur les trois jeux de données montrent que le modèle proposé a obtenu des performances compétitives par rapport à l'état de la technique, ce qui a permis de conclure à une meilleure représentation du document et à un meilleur résumé du document.

Nous trouvons aussi d'autres travaux qui ont implémenter deux algorithmes pour l'analyse de la complexité proposée par [Josef S & al, 2004] , le premier algorithme trouve le vecteur

de mot pour chaque mot qui ont représenté avec « *Word2Vec* » et « *Glove* » et ils le comparent aux vecteurs de chaque mot dans le document, tandis que le deuxième algorithme utilise le terme matrice de similarité de phrase calculée par le premier algorithme donc ce dernier est introduit comme entrée dans le deuxième algorithme.

5. Conclusion

Dans ce chapitre, nous avons dressé une lecture bibliographique sur les concepts et techniques de « *Deep Learning* » et « *Word Embedding* » ainsi que les différents outils et environnements ainsi que leurs implémentations dans les langages de programmation. Ensuite nous avons cité quelques travaux qui ont utilisé ces techniques pour la tâche d'indexation ou de recherches. On note aussi que nous avons intéressé dans ce chapitre sur les ressources textuelles comme des entrées de ces techniques parce que l'objectif de ce travail est de traiter les résultats de transcription des ressources parlées.

De point de vue technique, nous trouvons que l'algorithme « *Word2Vec* » est l'algorithme le plus utilisé dans sa version avec la bibliothèque « *Gensim* » pour le grand volume de données. Dans ce contexte, nous choisissons d'utiliser cet algorithme dans notre stratégie d'indexation sémantique.

Chapitre 3

Le modèle d'indexation sémantique proposé

1. Introduction

Dans ce chapitre, nous présentons une stratégie proposée pour l'indexation des ressources parlées transcrites par les APIs de reconnaissances en ligne proposées par Google. Dans ce contexte nous proposons d'utiliser le support de connaissance à base des systèmes d'apprentissage approfondi « *Deep Learning* » avec la technique « *Word Embedding* » en utilisons la bibliothèque « *Gensim* » sous le langage de programmation « Python » avec le modèle « *word2vec* » pour le processus d'indexation sémantique.

2. La Stratégie d'indexation sémantique proposée

L'exploitation des ressources parlées via leurs transcriptions automatiques par des systèmes de reconnaissance de la parole engendre plusieurs défis à traiter. En effet, les performances actuelles des systèmes disponibles de reconnaissances automatiques de la parole sont très satisfaisantes. En pratique, le développement des Interfaces Applicatives de Programmation « *Application Programming Interface* - API » dans le domaine de reconnaissance automatique de la parole ne cesse à progresser. Ces APIs sont des moyens efficaces pour faire communiquer nos scripts de reconnaissances avec des ressources locales ou dans le cloud, le tableau 3.1[1] ci-dessous présente quelques APIs pour la reconnaissance du contenu parlé qui se varient entre gratuites et payantes ainsi que leurs capacités de traitements.

Nom d'API	Description	Nature des traitements	Développeur	Langage	Licence
CLOUD SPEECH API	Speech to text conversion powered by machine learning	Online short utterance	Google	Over 80 Languages	Gratuit
Speechmatics API	cloud-based speech recognition based on the latest advances in deep learning neural networks	Online large files	Speechmatics	US and UK English	Payant
Bing Speech API	Convert audio to text, understand intent, and convert text back to speech for natural responsiveness.	Online short utterance	Microsoft	07 langages	Payant
API.AI	Natural language understanding platform	Online short utterance	api.ai	15 langages	Payant
Speech to Text API	Suite of language specific speech-to-text transcription software products by Vocapia Research for Linux x86 and x86-64 platforms.	Online large files	Vocapia Research	17 Langages	Payant
Speech Engine	A speech recognition engine, featuring a high recognition rate and built with the world leading speech recognition technology,	Online short utterance	iFLYTEK	Anglais et Chinois	Payant

TABLEAU 3.1 : LISTE DES APIs LES PLUS UTILISÉS POUR LA TRANSCRIPTION DU CONTENU PARLÉ.

[1] D'après le site : https://en.wikipedia.org/wiki/List_of_speech_recognition_software (DC : 23/05/2019)

Cependant, malgré les performances enregistrées dans ces systèmes de reconnaissances, nous trouvons les problèmes liés aux phénomènes des termes étranges de la langue utilisés et les termes techniques ou spécifiques ainsi que les termes composés ou complexes. Dans ce contexte, que nous essayons dans ce projet de fin d'études de proposer une stratégie d'indexation de ces ressources en utilisons la représentation vectorielle sémantique obtenue par le modèle « *Word Embedding* » et l'algorithme « *Word2Vec* » comme un support de connaissance pour l'indexation de ces ressources pour diminuer l'impact des insuffisances liées au système de reconnaissances automatiques utilisé pour la transcription du contenu des ressources parlées.

2.1. Philosophie de la stratégie proposée.

La tâche d'indexation des ressources textuelles est une tâche primordiale afin de les rendre exploitables par les systèmes informatiques. En pratique, nous trouvons des techniques d'indexation statistiques tels que le *tf-idf*. Cette technique permet de caractériser la pertinence de mots pour distinguer des textes pris dans un corpus. Cependant, nous trouvons d'autres techniques d'indexation qui intègre des métadonnées supplémentaires comme le « *Part-of-Speech - POS* » pour accentuer le processus d'indexation. Ainsi nous trouvons les travaux d'indexation sémantique basée sur le réseau sémantique « *WordNet* » en exploitant les différentes relations entre les mots et les groupes de mots « *Synset* ».

D'autre part, nous trouvons que la technique de « Word Embedding » offre une représentation vectorielle solide pour les relations sémantiques adjacentes des termes. Dans ce contexte que nous avons opté dans ce projet d'étudier la possibilité d'intégrer cette technique dans le processus d'indexation sémantique.

Notre stratégie proposée est composée de trois étapes :

- La Construction du modèle : dans cette étape, nous utilisons l'algorithme « *Word2Vec* » comme une implémentation de la technique « *Word Embedding* » pour la construction d'un modèle de représentation vectorielle des termes des ressources à indexer. Cette étape nous permet aussi d'avoir des modèles vectoriels dédiés au contexte ou topique spécifique si on veut construire des systèmes de recherches fermées ou dédiées à des domaines spécifiques.
- L'exploitation du modèle : dans cette étape, nous utilisons les fonctionnalités disponibles dans les modèles « *Word Embedding* » pour mesurer les similarités sémantiques existantes entre les termes susceptibles d'indexation. Dans cette phase, la sélection des termes d'indexation est effectuée à bases des scores de similarités obtenus entre les termes après les étapes de prétraitements usuelles tels que : « *Tokenezation* », « *Lemmatisation* » et « *Stemming* ».

- L'évaluation du modèle : dans cette dernière étape, nous avons utilisé un classifieur CNN avec la technique de « *Deep Learning* » pour évaluer les résultats d'indexation sémantique obtenus par notre modèle vectoriel définit par « *Word Embedding* ». En effet, l'utilisation de classifier CNN est fondée par la représentation matricielle (vectoriel) uniforme obtenue par l'algorithme « Word2Vect » pour chaque mot ou index retenu. Dans ce contexte, nous avons étudié la capacité de discrimination des termes d'index retenus via une tache de classification par topique ou domaine des ressources cibles.

2.2. Architecture du système d'indexation sémantique proposé

L'architecture proposée pour l'indexation sémantique de la transcription du contenu des ressources parlé est schématisée par la figure 3.1. Cette architecture décrit les différents composants nécessaires pour dérouler les différentes étapes d'indexation sémantique à partir de la représentation vectorielle des termes par la technique « *Word Embedding* ».

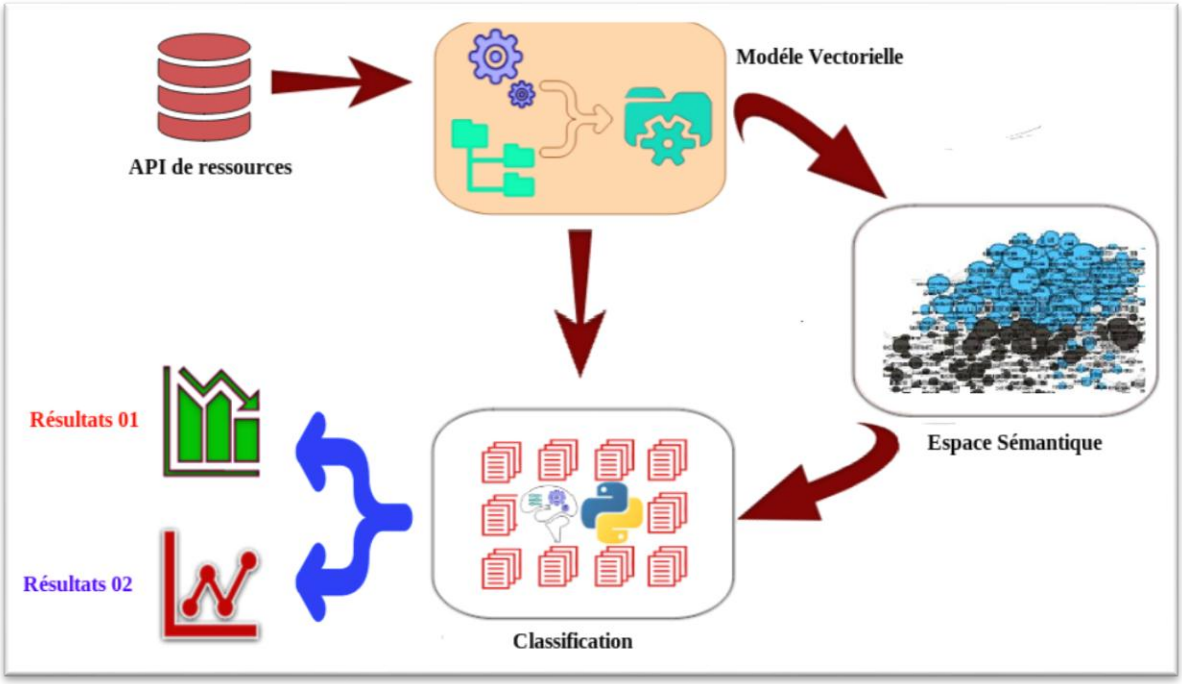

FIGURE 3.1 : STRATÉGIE PROPOSÉE POUR L'INDEXATION SEMANTIQUE.

3. Description de la stratégie

Dans cette section nous mettons l'accent sur les différents composants de l'architecture proposée en définissons les techniques et les outils nécessaires pour le processus d'indexation sémantique du contenu de transcriptions automatiques des ressources parlées avec la technique « Word Embedding ».

3.1. Le Passage du ressources parlées vers textuels

En effet, le passage de la primitive parole vers la primitive texte du contenu des ressources parlées est nécessaires pour qu'on puisse appliquer les différentes techniques d'indexation pour ces ressources. En revanche l'utilisation des systèmes de reconnaissances automatiques de la parole est la seule piste pour basculer de la primitive parole vers la primitive texte. Actuellement, nous trouvons des services en cloud tel que « *Google Cloud Speech API* », *« Pocket Sphynx* » qui permettent de réaliser cette tâche. Notons aussi que dans ce travail, nous n'avons pas focalisé sur cette étape, mais nous intéressons sur l'impact de cette phase sur la qualité des primitifs textes obtenus ainsi que les problèmes engendrés lors de la phase de transcription automatique.

3.2. La construction du modèle

Dans cette étape nous utilisons l'approche « *Word Embedding* » pour générer un modèle « *Word2Vec* » de notre ressource parlée. Avant de générer notre modèle, nous exécutons plusieurs étapes de prétraitements pour le processus d'indexation sémantique comme la présente la figure 3.2. En premier temps, nous utilisons les techniques de tokenisation en exploitons la bibliothèque *NLTK* sous *Python* avec la classe « *RegexpTokenizer* » pour traiter et segmentée nos ressources textuelles obtenues par l'étape précédente vers des mots ou techniquement des « Tokens ».

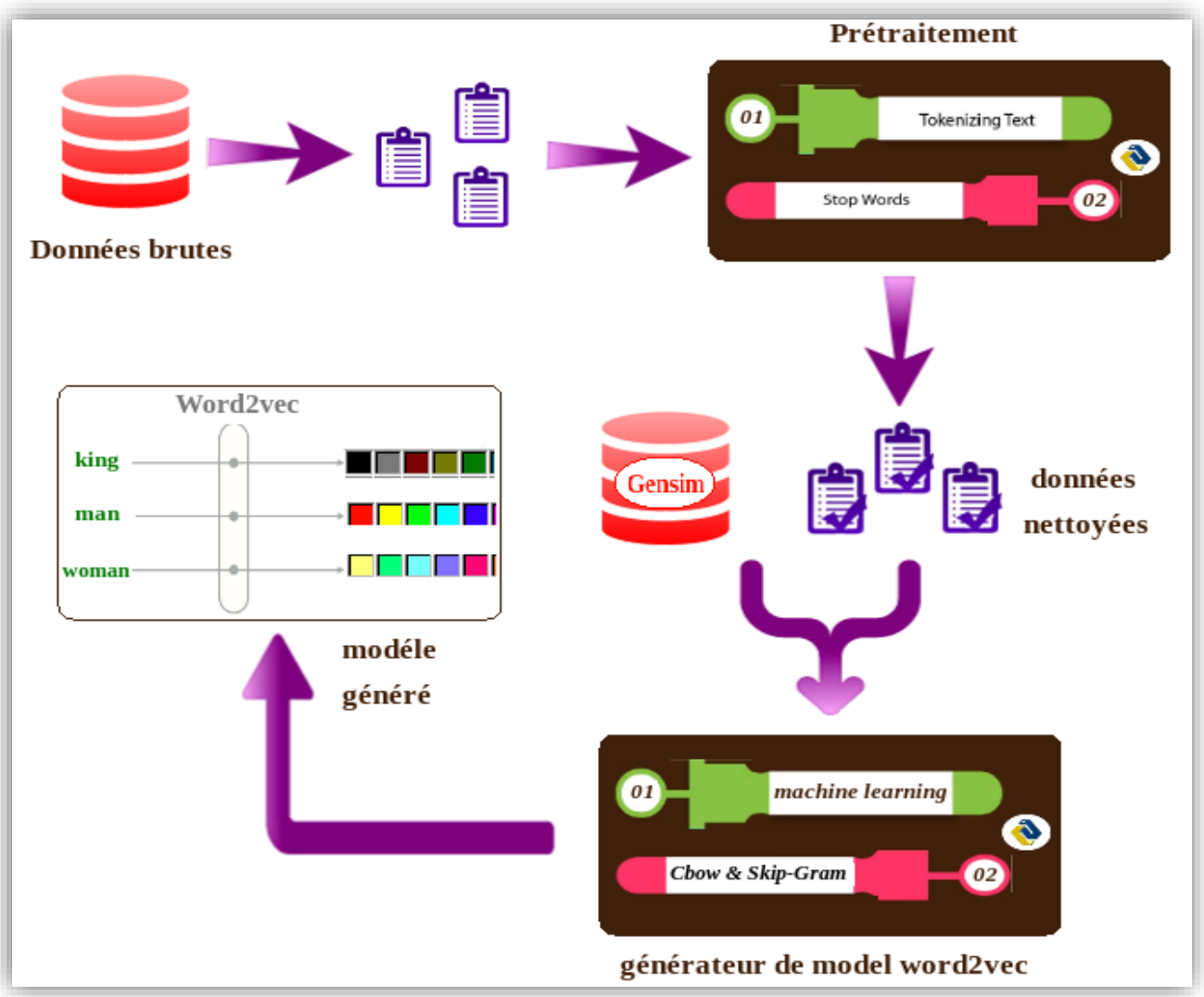

FIGURE 3.2 : LES ÉTAPES DE CRÉATION DU MODÈLE VECTORIEL.

Le résultat de cette tâche est une structure de type liste contient les termes ou « *Tokens* » qui seront utilisée pour l'apprentissage et la génération du modèle à l'aide des techniques de « Deep Learning » avec l'implémentation « *Word2Vec* » de « *Word Embedding* » comme la présente la figure 3.3. Notons aussi que dans cette phase, nous avons aussi étudié l'impact des termes vides ou les termes usuels dans les performances du modèle vectoriel obtenu.

```
>>> from nltk.tokenize import RegexpTokenizer
>>> tokenizer = RegexpTokenizer("[\w']+")
>>> sentence=tokenizer.tokenize("the deep learning is a part of machine
learning and I cant do without it ")
['the', 'deep', 'learning', 'is', 'a', 'part', 'of', 'machine', 'learning',
'and', 'I', "can't", 'do', 'without', 'it']
```

FIGURE 3.3 : EXTRAIT DE LA TÉCHNIQUE UTILISÉE POUR LA TOKENISATION.

Nous avons généré deux modules : le premier par l'exécution du processus d'apprentissage sur la totalité des ressources textuelle et le deuxième après une phase de prétraitement qui élimine les mots vides et les mots usuels de langue cible des ressources utilisés avec la bibliothèque NLTK implémentée sous *Python*. La figure 3.4 présente un exemple des routines python utilisé pour cette tâche.

```
>>> from nltk.corpus import stopwords
>>> stop_words = set(stopwords.words('english'))
>>>sentence1=[w for w in sentence if not w in stop_words ]
['deep', 'learning', 'part', 'machine', 'learning', 'I', "can't", 'without']
```

FIGURE 3.4 : EXTRAIT DE LA TÉCHNIQUE UTILISÉE POUR L'ÉLIMINATION DES MOTS VIDES.

Pour la génération et l'apprentissage de modèle « Word Embedding » nous recourrons à la bibliothèque de « *Gensim* » implémenté sous *Python*. En effet, sous *Python*, il existe l'implémentation des bibliothèques les plus populaires de « *Word Embedding* » telles que : *Keras*, *TensorFlow*, etc., mais nous avons opté pour ce choix pour la disponibilité du modèle linguistique adjacent ainsi que les ressources mémoires nécessaires pour l'exécution.

Nous avons utilisé le modèle « *Word2Vec* » de *Gensim* avec des paramètres sélectionnés qui permet d'ajuster les sorties d'apprentissage automatique pour obtenir la représentation vectorielle de chaque mot ou « *Tokens* ». En pratique, ce modèle utilise les paramètres suivants :

- **iter**(default=1): Numéro d'itération à exécuter lors de la phase d'apprentissage profond « epochs ».
- **size** :la taille de la représentation vectorielle du sortie,
- **sentence** : la structure qui contient les termes de ressources utilisées pour l'apprentissage du modèle.
- **hs** ({0, 1}, optional) : Si 1, la fonction d'activation dans le cas de l'approche CBOW
- **sg** ({0, 1}, l'approche utilisée pour le « *Word Embedding* » : *CBOW* ou *Skip Gram*.
- **window** (int, optional) : la taille de la fenêtre utilisée pour le « *Word Emedding* » pour la détection du contexte.
- **min_count** (int, optional) : Ignore tous les mots dont la fréquence totale est inférieure à celle-ci.
- **workers** (*int, optional*) : Utilisez pour paramétrer l'exécution de l'apprentissage pour l'exécution en parallèle selon les performances de la machine utilisée.

Dans la figure 3.5, nous présentons les paramètres utilisés pour générer notre modèle « *Word Embedding* » par la bibliothèque de *Gensim*

```
model=gensim.models.Word2Vec(sentence,min_count=1,size=300,iter=10,window=3,
sg=0,hs=1,workers=multiprocessing.cpu_count())
```

FIGURE 3.5 : PARAMETRES DU MODÈLE « WORD EMBEDDING » UTILISÉ

Le résultat de cette phase est un modèle contenant les représentations vectorielles de dimension 300 pour la totalité des termes des ressources utilisées comme il est représenté dans la figure 3.6 ci-dessous. On peut afficher un exemple comme suite :

```
>>>model.wv[deep]
[-0.06064018  0.0447381  -0.01000291  0.04606148  0.041887  -0.1124279...]
```

FIGURE 3.6 : EXEMPLE D'UNE REPRÉSENTATION VECTORIELLE D'UN MOT VIA LE MODÈLE GÉNÉRÉ.

Finalement, nous devons sauvegarder le modèle généré de nos ressources utilisées sous le format « Word2Vec » dans un fichier texte pour son utilisation dans les étapes suivante.

3.3. L'exploitation du modèle – Indexation sémantique -

Dans cette section, nous exploitons le modèle développer dans la section précédente comme un support d'indexation sémantique des ressources utilisées. Dans ce contexte, nous utilisons les clauses de similarités définies dans les modèles « *Word Embedding* » et « *Word2Vec* » pour la détection des termes d'indexation les plus discriminants. Aussi l'utilisation de la représentation vectorielle des termes de ressources ainsi que des relations sémantiques inter terme pour trouver les termes les plus représentatifs des concepts existant dans le contenu des ressources étudiées. Les relations sémantiques utilisées dans notre modèle sont les similarités fournies par l'apprentissage de notre modèle via les techniques d'apprentissage « *Deep Learning* ». La figure 3.7 décrit les différentes étapes pour l'indexation sémantique du contenu des ressources à l'aide de l'algorithme « *Word2Vec* » de la représentation vectorielle « *Word Embedding* ». Dans les sections qui se suit nous détaillons les descriptions de chaque étape.

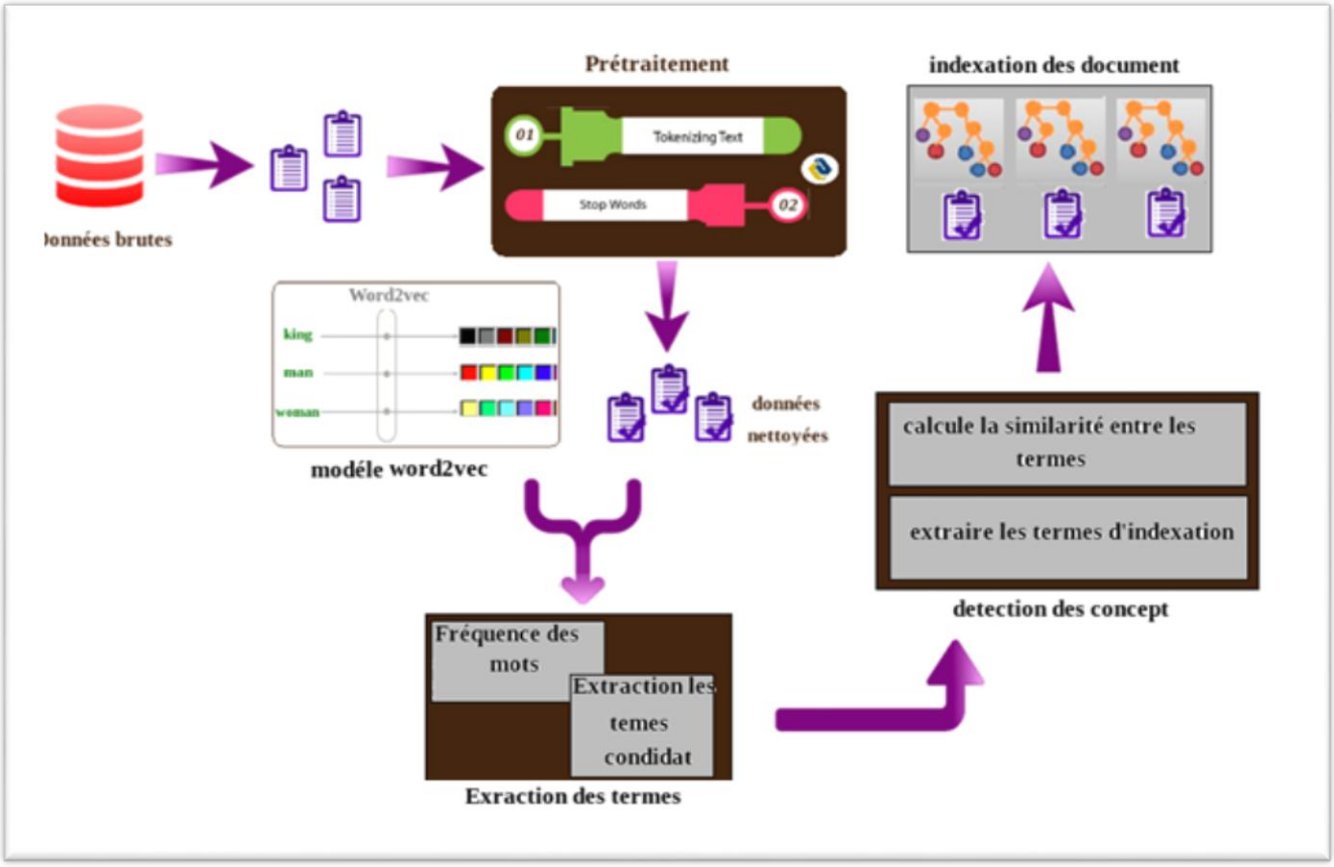

FIGURE 3.7 : INDEXATION SÉMANTIQUE DES RESSOURCES TEXTUELLES.

3.3.1. Les prétraitements

Dans cette étape, nous récurons aux mêmes prétraitements de notre donnée avec l'exécution les mêmes étapes de prétraitement utiliser dans l'étape de création du modèle pour le calcul et la détection des similarités entre les termes des ressources utilisées susceptibles d'indexation.

3.3.2. Calcul de la similarité

Après l'étape de prétraitement, nous utilisons notre modèle « word2vec » généré dans la première étape avec la bibliothèque *Gensim* avec l'instruction présentée dans la figure 3.8 afin de l'utiliser dans nos routines offlines :

```
>>> model = KeyedVectors.load_word2vec_format("all2.txt")
```

FIGURE 3.8 : CHARGEMENT DE NOTRE MODÈLE VECTORIEL.

Ainsi, nous recourrons aux calculs de similarités binaires entre le terme. En effet, le calcul intégral des similarités binaire est quasiment impossible pour les termes. Pour cela, nous utilisons les fréquences d'apparitions des termes pour délimiter le nombre de combinaisons. De point de vue, métriques de similarités, nous trouvons plusieurs méthodes dans « Gensim » et « word2vec » qui permets de calculer ces similarités, parmi eux nous citons :

- *most_similar(word)* :Cette méthode calcule la similarité en utilisons une moyenne simple des vecteurs de pondération de projection du mot en entrée avec la représentation vectorielle des mots du modèle. Cette similarité est paramétrée par :
  - *positive* ( ) - Liste de mots les plus similaires positivement.
  - *negative* ( ) - Liste de mots les plus similaires négativement.
  - *topn* ( ) - Nombre de N premiers mots similaires à renvoyer.
  - *restrict_vocab* ( ) - Entier optionnel qui limite la plage de vecteurs recherchés pour les valeurs les plus similaires. Par exemple, *restrict_vocab* = 10000 ne vérifiera que les 10000 premiers vecteurs de mots dans l'ordre du vocabulaire. Par exemple :

- similarity(word$_1$, *word*$_2$)
  Cette méthode permet de calculer la similarité entre deux mots *word*$_1$ et *word*$_2$ par une valeur numérique comprise entre 0 et 1, la figure 3.9 présente un exemple de cette similarité.

```
>>> model.similarity('ted','conference')
0.555557
```

FIGURE 3.9 : EXEMPLE DE CALCUL DE SIMILARITÉ.

3.3.3. Indexation sémantique

La recherche des similarités binaires pour tous les termes d'une ressource est fastidieuse. Pour cela, nous utilisons la technique *tf* ; fréquence de termes ; d'indexation utilisée dans le modèle vectoriel pour diminuer le nombre de termes susceptibles d'indexation. Pour cela, nous gardons que les termes qui ont une fréquence d'apparition comprise entre l'intervalle [*S_freq_min*, *S_freq_max*]. Cependant, pour le traitement d'impact des processus de transcription sur la qualité des ressources textuelles obtenues, nous utilisons la représentation vectorielle par le modèle obtenu « *Word Embedding* » pour deux vocations. La première pour filtrer et garder seulement les termes d'indexation pertinents avec la classe « *similarity* » de « *Word2Vec* ». Entre temps, nous recourons à un processus d'enrichissement de ces termes par la classe « *most_similar* » de « *Word2Vec* ». La figure 3.10 présente une description pragmatique de l'algorithme utilisé dans cette étape.

```
1.Calculation de matrice des fréquences de chaque terme.
2. Initialisation des seuils de fréquences d'apparitions
S_freq_min # Seuil de fréquence minimum
S_freq_max # Seuil de fréquence maximum
Seuil
 # Seuil minimum de similarités
Term_Index= {}
3. Construction de l'ensemble des termes candidats d'indexation
Term_Index_condidats =
If (freq_W>= S_freq_min and freq_W<= S_freq_max
Term_Index_condidats.insert (W)
4. Calcul des similarités sémantiques
for each Wi in Term_Index_condidats
for each (Wk in Term_Index_condidats) and (i<>j)
SimW2V(i,j) = Mesure_similarity(Wi,Wj)
5. Détection des termes similaires
Similar_Terms ={}
for i=1 : nb_Words
for j=1 : nb_Words -1
if (i<>j) and (abs(SimW2V(i,j)- SimW2V (i,j+1))>= Seuil)
Similar_Terms.add(Wi,Wj)
Term_Index.insert (Wi,freq_Wi)
Term_Index.insert (Wj,freq_Wj)
```

FIGURE 3.10 : ALGORITHME D'INDEXATION SÉMANTIQUE.

3.4. Évaluation du modèle

Généralement l'évaluation des stratégies d'indexation s'effectue par l'évaluation des systèmes de recherche d'information qui les utilisent. Cependant, l'objectif de ce travail est l'intégration des techniques de « *Word Embedding* » dans le processus d'indexation sémantique. Entre temps, le développement d'un système de recherches d'informations qui intègre cette stratégie d'indexation reste parmi nos objectifs futurs. Dans ce contexte, nous proposons dans ce mémoire d'utiliser un classifieur « *CNN* » pour évaluer l'impact de la stratégie d'indexation. Pour cela, nous avons conçu un classifieur pour la classification à base du contenu intégral des transcriptions des ressources utilisées. L'objectif est de maximiser les performances de note classifieur « *CCN* ». Ainsi, nous utilisons ce classifieur pour la classification des ressources à base de l'ensemble des termes d'indexation détecter dans l'étape précédente.

3.4.1. Architecture proposée du classifier « *CNN* »

L'architecture utilisée dans notre classifieur contient trois couches. La première couche de notre modèle de classification c'est la couche d'*Embedding* basée sur la représentation vectorielle fournie par le modèle « *word2vec* ». Tandis que, pour la deuxième nous proposons une couche de convolution *1D*. Cependant pour la couche de sortie, nous utilisons une couche entièrement connectée « *Fully_Connected* » qui permet de calculer le vecteur de classification. L'architecture de notre classifieur proposée est schématisée par la figure 3.11.

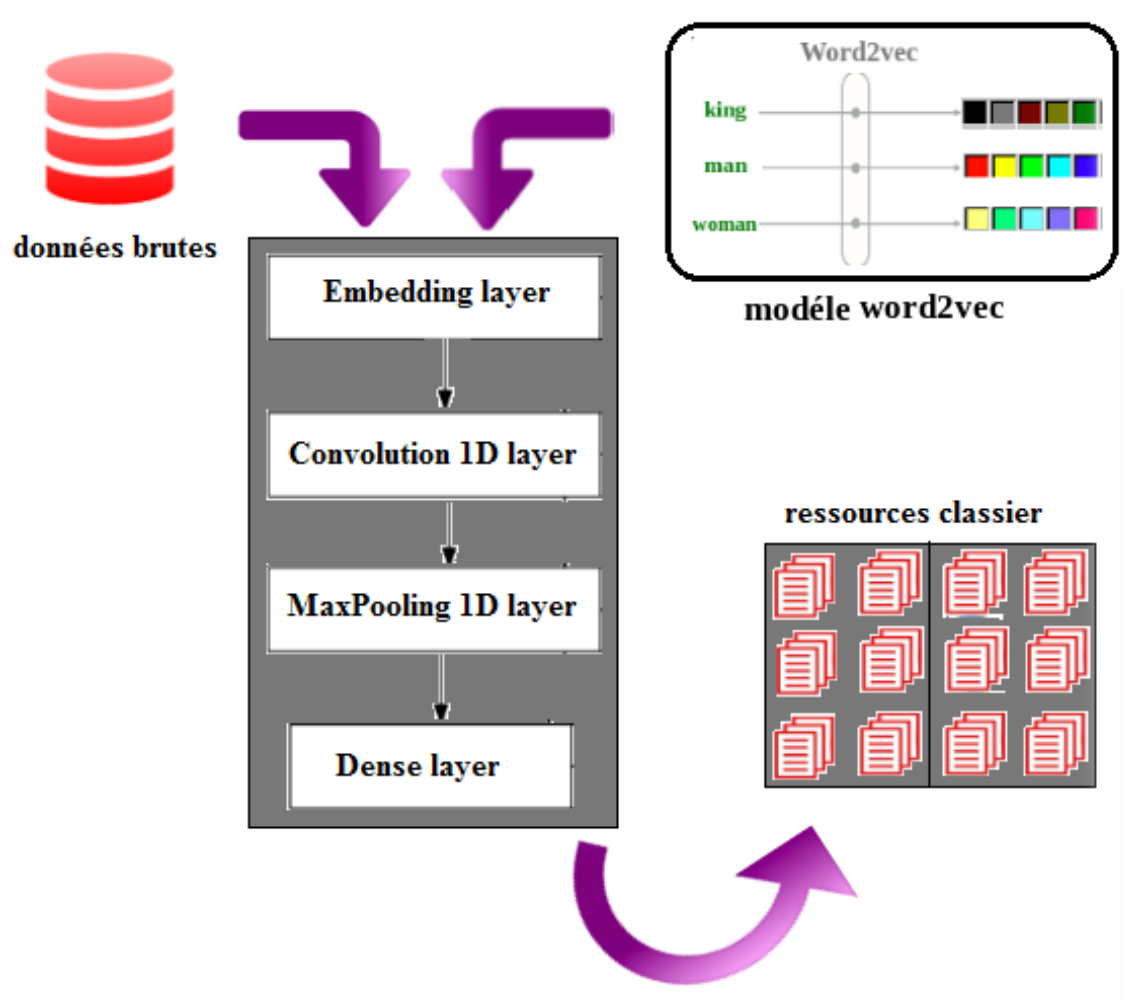

FIGURE 3.11 : ARCHITECTURE DE CLASSIFIEUR CNN PROPOSÉE

De point de vue pratique, nous avons utilisé un corpus de cinq classes ; qui sera détaillé dans le chapitre suivant ; ainsi, que pour l'apprentissage et l'évaluation du classifieur CNN proposée nous avons étiquetés les transcriptions textuelles de nos ressources avec un vecteur binaire. Ensuite nous avons divisé le corpus en deux parties : Train et Test. La représentation vectorielle de ces ressources est réalisée par le modèle « Word Embedding » avec l'algorithme « Word2Vec » avec la bibliothèque « Gensim ». En fin, la couche « *Fully Connected* » fournit une sortie de dimension cinq qui représente le nombre de classes de notre système. L'architecture est détaillée dans la figure 3.12 ci-dessous :

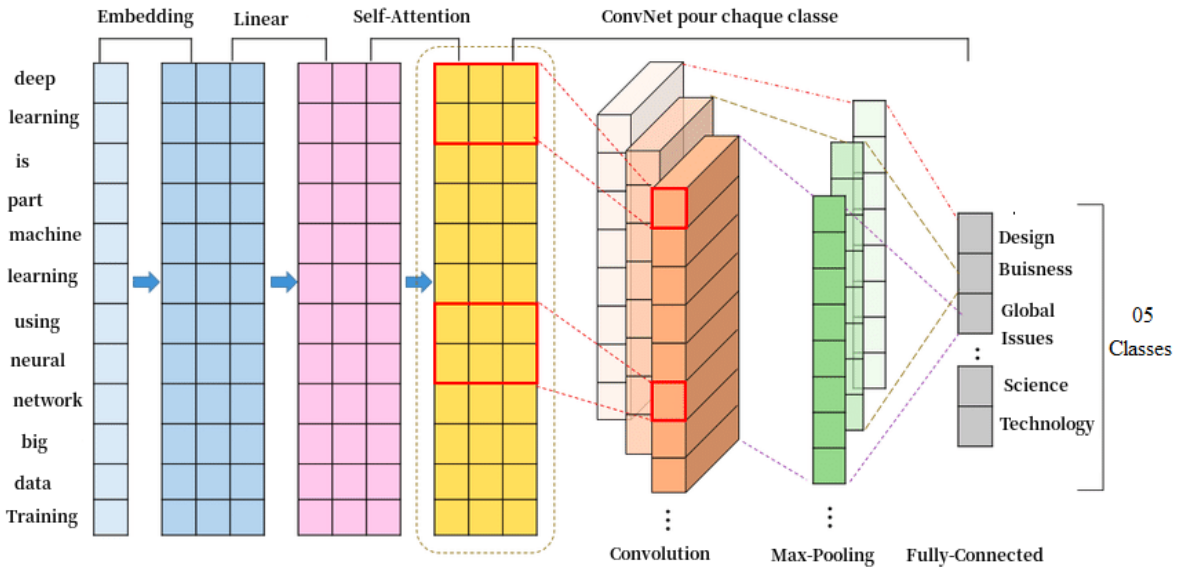

FIGURE 3.12 : L'ARCHITECTURE DÉTAILLÉE DU CLASSIFIER AVEC UNE COUCHE EMBEDDING.

3.4.2. Implémentation du classifier « *CNN* » proposé

Pour la phase de mise on œuvre de ce classifieur, nous avons utilisé la bibliothèque *« Keras »* sous *Python*. Pour le classifieur nous avons opté pour un modèle séquentiel avec une pile linéaire avec l'empilement de trois couches telles que :

- Une couche *Embedding* dans sa taille qui représente les termes « Tokens » en entrée
- Une couche de convolution de 150 filtres en 3 dimensions avec une fonction d'activation telle que : *ReLU*, *SoftMax* et *maxpooling* de taille 2
- Une *FC* qui va créer le vecteur final à envoyer au réseau de neurones artificiels ainsi qu'une fonction d'activation « *Sigmoid* » pour les cinq « *RNA* » de sortie

Les figures 3.13 et 3.14 décrit les paramètres utilisés dans notre classifieur « *CNN* »proposé.

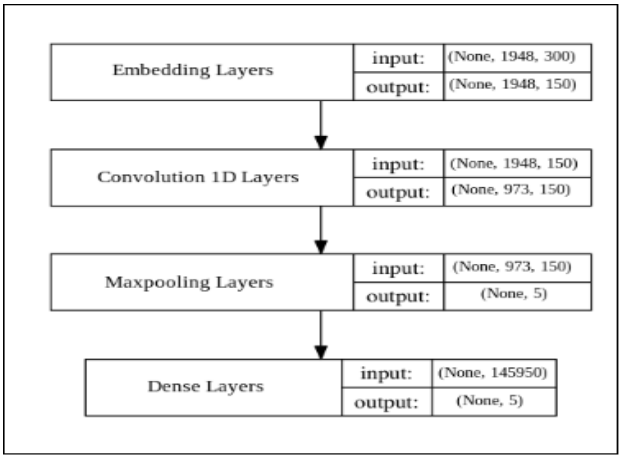

FIGURE 3.13 : REPRÉSENTATION DES COUCHES ET DES PARAMÈTRES DE NOTRE CLASSIFIER.

```
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
classifier.add(embedding_layer)
classifier.add(Conv1D(150, kernel_size=3, activation=act))
classifier.add(MaxPooling1D(pool_size=2))
classifier.add(Flatten())
classifier.add(Dense(5, activation='sigmoid'))
```

FIGURE 3.14 : LES COUCHES AVEC LES PARAMÈTRES DE NOTRE CLASSIFIER.

Pour l'apprentissage du classifieur proposé, nous avons utilisé les paramètres d'apprentissage suivants :

- L'erreur « *loss* » : c'est le paramètre qui permet de mesurer l'écart entre les prédictions et les résultats « $Y - \widehat{Y}$ ». Nous avons utilisé la fonction « *cross entropy* ».
- La Rétropropagation d'erreur « *l'optimizer* » : c'est un algorithme qui permet la mise à jour des poids pour les valeurs des poids du modèle. Nous avons utilisé les algorithmes de « *Adam* » et « *Descent du Gradient* »

La figure 3.15 suivante, présente un exemple des paramètres utilisé pour l'apprentissage

```
classifier.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

FIGURE 3.15 : LES PARAMÈTRES UTILISÉS POUR L'APPRENTISSAGE.

En résumons, la figure 3.16 présente les différentes étapes utilisées pour la définition de l'architecture de classifieur « CNN » proposé ainsi que les différents paramètres et fonction utilisée pour la création de ce classifieur.

```
1- étiquitage de chaque documents
2- selection des document :
train_docs# données d'entraiment
test_docs# données de test
3-chargement de  model word2vec se formes d'une list;
model=list(model)
4-pretraitement de train_docs , test_docs ;
train_list=clean_docs(train_docs,model)
test_list=cean_docs(test_docs,model)
5-Initialisation de:
Ytrain#label de entraiment
ytest label de test
for each lines in train_docs
ytrain_concate(lines[0])
for each lines in trest_docs
ytest_concate(lines[0])
6-indexation des mots par la position;
index_train(train_list)
index_test(test_list)
7-initialisation de :
xtrain
xtest
for each w1,w2 in train_list,test_list
xtrain_concat(index_train[w1])
xtest_concat(index_test[w2])
8-charger le model word2vec format embedding;
for each in in model_size
embedding(model.vecteur[i])
9-creation de matrice embedding de tous les mots avec le vrais position
matrice_embedding(model)
10-creation les couche de classifier
couche_embedding
couche_convolution_1D
couche_fully_connect
11-formation de classifier
```

FIGURE 3.16 : LES ÉTAPES UTILISÉES POUR LE CLASSIFIEUR PROPOSÉ.

4. Conclusion

Nous avons présenté dans ce chapitre notre stratégie proposée pour l'indexation sémantique du contenu des transcriptions de ressources parlé avec l'intégration les techniques de « *Word Embedding* » et « *Word2Vec* » comme un support de représentation de connaissances. Cependant, pour mesurer l'impact de cette stratégie d'indexation sémantique, nous avons proposé l'utilisation d'un classifieur « *Deep Learning* » pour l'évaluation des performances de notre stratégie proposée. Dans le chapitre suivant, nous présentons les démarches utilisées pour la validation ainsi que les divers résultats obtenus.

Chapitre 4

Tests d'expérimentations et évaluation

1. Introduction

Après avoir étudié dans les parties précédentes les différents outils du *« Deep Learning »* et *« Word Embedding »* ainsi que son algorithme d'implémentation « Word2Vec », nous allons présenter dans cette partie une démarche da validation de notre stratégie par des tests d'évaluation du modèle proposé et développé. Cependant pour évaluer les performances de notre stratégie d'indexation sémantique à base de « Word2Vect », nous exécutons une série de tests sur notre classifieur « CNN » proposé.

2. Ressources utilisées

Nous avons intéressé à la problématique d'indexation sémantique du contenu de la transcription des ressources parlées en utilisons la technique *« Word Embedding »* pour une représentation vectorielles de leurs contenus ainsi que les différentes similarités sémantiques qui les intègrent. Plusieurs bases ont été présentées dans la littérature, cependant la grande majorité de ces ressources sont payantes et sous la licence de NIST[1] . A cet effet, nous avons choisis d'utiliser un extrait du corpus « *TEDLIUM V1*[2] » réalisé à partir des ressources de la communauté de « *TED Talks* [3]». Nous avons utilisé un extrait de cette base qui se compose d'un ensemble de ressources parlées pour l'apprentissage et le test. Ainsi, nous avons étiqueté ces ressources sous cinq classes pour l'apprentissage. On note aussi que ces classes ne sont pas linéairement séparable. Donc, nous pouvons trouvez une source dans deux ou même trois classes différentes. Cette complexité est très utile et surtout pour l'évaluation de la capacité de la discrimination de processus d'indexation sémantique proposé.

3. Présentation des outils utilisés

3.1. Le software

Plusieurs Framework et bibliothèques open sources sont disponibles dans la littérature, la grande majorité supporte le langage Python. Nous avons utilisé le langage *python 2.7* et une listes de quelques bibliothèques de « *Machine Learning* » préparé avec des paramètres :

3.1.1. La bibliothèque Gensim

La bibliothèque « Gensim » est souvent utilisée pour les techniques de *« Word Embedding »* Nous avons utilisé cette bibliothèque dans sa version « *Gensim 3.7.1* », pour crée le modèle « word2vec » et pour les traitements sémantiques adjacentes.

[1] https://www.nist.gov/
[2] http://www-lium.univ-lemans.fr/en/content/ted-lium-corpus
[3] https://www.ted.com/talks

- Avantages
    - Libre pour les développeurs sous *Python*.
    - Disponible pour tous les systèmes d'exploitation des ordinateurs : Pc, Mac.
    - Très recofondé pour les techniques de « Latent Semantic Analyses ».
- Inconvénient
    - Nécessite un large espace en mémoire des calculs.

3.1.2. La bibliothèque Keras

« Keras » représente la bibliothèque la plus sollicitée pour la création des classifieurs « *CNN* ». Elle est souvent utilisée en collaboration avec les bibliothèques « *Tensorflow* » et « *Theano* ». Dans notre système, nous avons utilisé cette bibliothèque dans sa version « *Keras 2.2.4* », pour la création du modèle de classification « *CNN* ».

- Avantages
    - Open sources pour Python.
    - Son intégrité avec « *Tensorflow* » ou « *Theano* ».
    - Interface haut niveau, intuitive.
- Inconvénient
    - Très gourmande de point de vue calculs et espace mémoire.

3.1.3. La bibliothèque Tensorflow

Dans le deuxième chapitre, nous avons mentionné que « *TensorFlow* » est une bibliothèque unitaire et complète crée par l'équipe de Google Brain orienté vers les travaux de recherches du domaine *« Machine Learning* ». Nous avons choisi d'utiliser « *Tensorflow* » comme un *backend* pour la bibliothèque « *Keras* ».

- Avantages
    - Open sources pour Python.
    - Très recofondé pour les techniques de « Latent Semantic Analyses ».
    - Supporte plusieurs modes de calculs : GPU, CPU et TPU.
- Inconvénient
    - Plus lente que les autres Framework « *Theano* ».
    - Le support des *RNN* est encore surclassé par « *Theano* ».

3.2. Le hardware

Le « *Deep Learning* » est un domaine avec des exigences des calculs intenses et la disponibilité des ressources (surtout GPU) qui vont fondamentalement influencer sur l'expérience de l'utilisateur car sans ses ressources, il faudra trop de temps pour la construction des modèles ainsi la détection des erreurs de programmations adjacentes, ce qui rend la tâche de programmation très lourdes.

Les expérimentations ont tous été effectuées sur une machine qui offre des performances acceptables dont voici les caractéristiques utiliser dans notre travaille :

CPU	Intel® Core™ i3-4005U (1.70 GHz)
GPU	NONE
RAM	4 GO

TABLEAU 4.1 : LES PERFORMANCES DE NOTRE MACHINE.

Cependant, vue les fortes exigences de calculs et espaces nécessaires pour l'implémentation des solutions à base de « TensorFlow » et « Keras », nous avons opté pour l'utilisation de calculs en *Cloud* via la plateforme « *Google Colab* ». Cette dernière offre la possibilité de calcul *GPU, TPU* et même *CPU.* C'est une plateforme gratuite pour des sessions de temps limitées sur le web. Elle utilise l'environnement « *Jupyter Notebook* » pour les différentes routines de programmation en langage Python.

4. Validation et résultats

Dans cette section nous présentons la démarche utilisée pour la validation des différentes étapes de notre stratégie d'indexation sémantique proposée. Ainsi, nous discutons les résultats obtenus par notre travail, technique d'intégration des mots *« Word Embeddings »* pour l'indexation sémantique du contenu de transcription des ressources parlées et leur classification.

4.1. Génération le modèle

Dans cette section, nous avons fait une étude comparative sur les deux modèles « *word2vec* » connues « CBOW » et « Skip-Gram » avec des différents paramètres et pour générer notre modèle vectoriel, qui nous fournit des différents résultats selon le changement des paramètres « word2vec ».

Le tableau représente les résultats obtenus :

Paramètres / Model	Itérations	Temp de création le modèle (S)	Taille de modèle sortie (MB)
CBOW	5	10.07	53.4
Skip-Gram	5	23.26	52.4
CBOW	10	20.02	51.7
Skip-Gram	10	48.87	51.3
CBOW	20	34.57	50.2
Skip-Gram	20	92.89	50.4
CBOW	30	53.22	49.5
Skip-Gram	30	138.27	50.1

TABLEAU 4.2: LES DIFFÉRENTS MODÈLES WORD2VEC.

A partir de la deuxième partie nous mentionons que le modèle « CBOW » qui utilise les mots de contexte pour produire la prédiction et la représentation de mot centrale mais le modèle « Skip-Gram » utilise l'inverse du 1er modèle, ce qui affecte la durée de traitement et les couches des réseaux de neurones artificiels, qui vont expliquer que le temps de construction de modèle « CBOW » est inferieur par rapport au temps de construction de modèle « Skip-Gram ».

L'efficacité de chaque modèle a été mesurée avec des étapes de test sur le calcul de similarité, nous avons mentionné quelques résultats de test de chaque étapes montrée dans le tableaux précédent, Après l'exécution de l'instruction ***« model.most_similar('talk')»*** pour afficher les mots les plus similaires à « talk » nous avons trouvé les résultats de chaque étape comme est représenté dans la figure 4.1 ci-dessous :

Iteration=5, CBOW

```
[('talking', 0.7738348245620728),
 ('worry', 0.7381659746170044),
 ('debate', 0.7187763452529907),
 ('euros', 0.7093985080718994),
 ('eliminating',
0.7043022513389587),
 ('talked', 0.696002721786499),
 ('tell', 0.6802690625190735),
 ('thinking', 0.6726889610290527),
 ('think', 0.6706113219261169),
 ('launch', 0.6687730550765991)]
```

Iteration=5, Skip-Gram

```
[('ambivalently',
0.7014768719673157),
 ('adopting', 0.6536180377006531),
 ('curse', 0.6473675966262817),
 ('fulfill', 0.6328582763671875),
 ('burdening', 0.6282455921173096),
 ('classify', 0.6231489777565002),
 ('ought', 0.6198402643203735),
 ('varied', 0.6176761984825134),
 ('talking', 0.6175950765609741),
 ('opinions', 0.6163890361785889)]
```

Iteration=10, CBOW	Iteration=10, Skip-Gram
[('talking', 0.6291012763977051), ('worry', 0.592960774898529), ('talked', 0.5813649892807007), ('depress', 0.5735804438591003), ('think', 0.5521660447120667), ('launch', 0.5442180633544922), ('euros', 0.5270152688026428), ('thinking', 0.5235194563865662), ('eliminating', 0.4996328055858612), ('concern', 0.4993358254432678)]	[('ambivalently', 0.5311927795410156), ('curse', 0.48767730593681335), ('fulfill', 0.4808354377746582), ('depress', 0.4733307361602783), ('classify', 0.4721135199069977), ('adopting', 0.4607574939727783), ('about', 0.45926839113235474), ('varied', 0.4590522050857544), ('observations', 0.45744162797927856), ('themes', 0.4464235305786133)]
Iteration=20, CBOW	**Iteration=20, Skip-Gram**
[('worry', 0.41444969177246094), ('talking', 0.3914059102535248), ('concerns', 0.3897296190261841), ('thinking', 0.3828887343406677), ('depress', 0.38285398483276367), ('think', 0.3826827108860016), ('talked', 0.37585824728012085), ('tell', 0.3643950819969177), ('cartography', 0.3591158390045166), ('understand', 0.3535670042037964)]	[('ambivalently', 0.4344635605812073), ('curse', 0.39524155855178833), ('about', 0.3846326172351837), ('willie', 0.3813992738723755), ('smits', 0.37469807267189026), ('adjudicate', 0.3697926700115204), ('adopting', 0.362055242061615), ('varied', 0.3499947190284729), ('classify', 0.3498581051826477), ('cartography', 0.3471609055995941)]
Iteration=30, CBOW	**Iteration=30, Skip-gram**
[('talking', 0.36859896779060364), ('cartography', 0.3631618916988373), ('thinking', 0.35932427644729614), ('talked', 0.3482329249382019), ('depress', 0.34521329402923584), ('worry', 0.34179139137268066), ('complain', 0.3275268077850342), ('think', 0.3141782581806183), ('daydream', 0.3094485402107239), ('concerns', 0.3047178387641907)]	[('ambivalently', 0.4058876633644104), ('willie', 0.3989964723587036), ('about', 0.39220765233039856), ('smits', 0.3870370090007782), ('curse', 0.37484288215637207), ('classify', 0.33439621329307556), ('defray', 0.3303966224193573), ('foresting', 0.326246976852417), ('firsthand', 0.3235732913017273), ('linguistics', 0.3223685622215271)]

FIGURE 4.1 : LES MOTS LES PLUS SIMILAIRES À « TALK » DANS DIFFÉRENTS MODÈLES.

Les résultats montrent que la similarité a changé en fonction des itérations et du type de modèle utilisé ,où le degré de similarité diminue à chaque fois quand on augmente le nombre des itérations d'apprentissage par modèle, Par exemple les résultats relatifs aux plus similaires à « talk », avec l'utilisation du modèle « CBOW » et l'apprentissage (des réseaux neurones) qui se fait avec 10 itérations. Les résultats obtenus sont très acceptables et exploitable par la tâche d'indexation sémantique.

4.2. Indexation sémantique

Il est connu que la tâche d'indexation sémantique a une grande importance et très intéressante aux chercheurs du domaine LSI, pour la création des différents systèmes de différents techniques pour améliorer ce domaine. Dans cette section on présente les différents statistiques obtenus par notre système sémantique.

Après l'application de notre système sur quelques documents, nous avons obtenu les résultats après le déroulement des étapes de test appliqué avec le changement de seuil de la similarité ; nous avons trouvé ces résultats :

Documents	Nombre des termes candidat (Kilo mots)	Nombre des termes d'indexation (Kilo mots)	Le pourcentage %
0.55 – 0.85	1257	158	12.56
0.55 – 0.80	1257	158	12.56
0.55 – 0.75	1257	156	12.41
0.55 - 0.70	1257	156	12.41
0.55 - 0.65	1257	151	12.01
0.55 - 0.60	1257	128	10.18
0.55 - 0.55	1257	0	0.0
0.50 – 0.85	1257	287	22.83
0.50 – 0.80	1257	287	22.83
0.50 – 0.75	1257	285	22.67
0.50 - 0.70	1257	285	22.67
0.50 - 0.65	1257	283	22.51
0.50 - 0.60	1257	270	21.47
0.50 - 0.55	1257	225	17.89
0.45 – 0.85	1257	466	37.07

0.45 – 0.80	1257	466	37.07
0.45 – 0.75	1257	465	36.99
0.45 - 0.70	1257	465	36.99
0.45 - 0.65	1257	465	36.99
0.45 - 0.60	1257	463	36.83
0.40 – 0.55	1257	450	35.79

TABLEAU 4.3 : LES DIFFÉRENTES STATISTIQUE D'INDEXATION SÉMANTIQUE.

Ce résultat obtenu montre que les termes d'indexation diminuent avec le changement d'intervalle de seuil de la similarité, ces termes obtenus avec notre système sont des termes représentatifs de contexte de chaque ressource. Cependant, on évalue la capacité de discrimination de ces termes retenus.

4.3. Classification des ressources

Afin d'évaluer les performances du modèle de classification proposé dans la section 3.4.1 du troisième chapitre, nous allons expérimenter le même modèle décrit par la figure 3.11. Il y a trois couches essentielles dans notre modèle de classification, la première c'est une couche *embedding*, ensuite une couche *convolutionnel* et en sortie une couche *fully connect*. La première couche utilise comme entrée une matrice *embedding* comme un noyau de taille contexte * 300, Les hypers paramètres de « A*dam*» ont initialisé à leurs valeurs par default. La taille du batch est par default, La fonction coût est l'entropie croisée fixée dans tous étapes par la formule :

$$H(p,s) = -\sum_i p\, log(s) \qquad (01)$$

- avec p la sortie désirée et s la sortie du réseau de neurone.

La couche convolutionnelle avec des entrées 150 et valeur de « kernel » de taille 5 plus de fonction d'activation à la fin, il y a aussi une couche *maxpooling* pour la couche précèdente de taille 2, après la couche *dense* ou fully connect pour l'organisation et la classification des entrées avec les cinq classes de sorties. Le nombre d'époque est testé sur 30 itérations. Les hypers paramètres de « Adam », la taille du batch est par default pour l'apprentissage et pour le test.

4.3.1. Préparation des données

Notre modèle a besoin d'une tache plus importante pour créer le « Y_train » et « Y_test », dans ce cas-là nous allons utiliser l'étiquetage des ressources, pour construire le « Y_train » et « X_train » qui permet d'orienter la tache de classification et les réseaux de neurones et

donne des bons résultats. Après l'application de l'algorithme d'étiquetage nous obtenons des résultats comme suit :

Document	Étiquetage
AlanSiegel_10.txt	[1, 0, 0, 0, 0]
KevinBales_10.txt	[1, 0, 1, 0, 0]
KevinStone.txt	[0, 1, 0, 0, 1]
BillGates_10.txt	[0, 0, 1, 1, 1]
NathlanWolfe_09.txt	[0, 0, 1, 1, 0]
RichardStJohn_09.txt	[1, 0, 0, 0, 0]
KevinSurace_09.txt	[1, 1, 0, 0, 1]

TABLEAU 4.4: ÉTIQUETAGE DES DOCUMENTS.

Comme nous avons vu l'algorithme d'étiquetage donne des lignes dans chaque ressource ce forme une liste de taille cinq pour les classes de sorties. Les résultats d'algorithme obtenus montrent que les ressources sont reparties dans plusieurs domaines ; par exemple la ressource *KevinSurace_09.txt* dans le tableau 4.4, qui existe dans les domaines 1, 2 et 5. Maintenant chaque document étiqueté, les résultats de cette tâche permet de créer « Y_train », Y_test. Qui guide le système de classification.

4.3.2. Apprentissage et évaluation du classfieur CNN

Après la création de notre modèle de classification les couple « Y_train » , « X_train » et « Y_test », « X_test », nous pouvons maintenant exécuter la tâche d'apprentissage sur notre modèle de classification avec des différentes étapes et faire une étude comparative entre les résultats.
Nous testons notre modèle de classification avec 2 types des fonctions d'activation tel que :

- ReLu
- Softmax
- Tanh

On débute avec des tests se font avec trois fonctions d'activation « Relu » , « softmax » et « Tanh », on fixe la fonction d'activation de la couche fully-connect à sigmoid (voir figure 4.2). Voici le résultat dans le graphe :

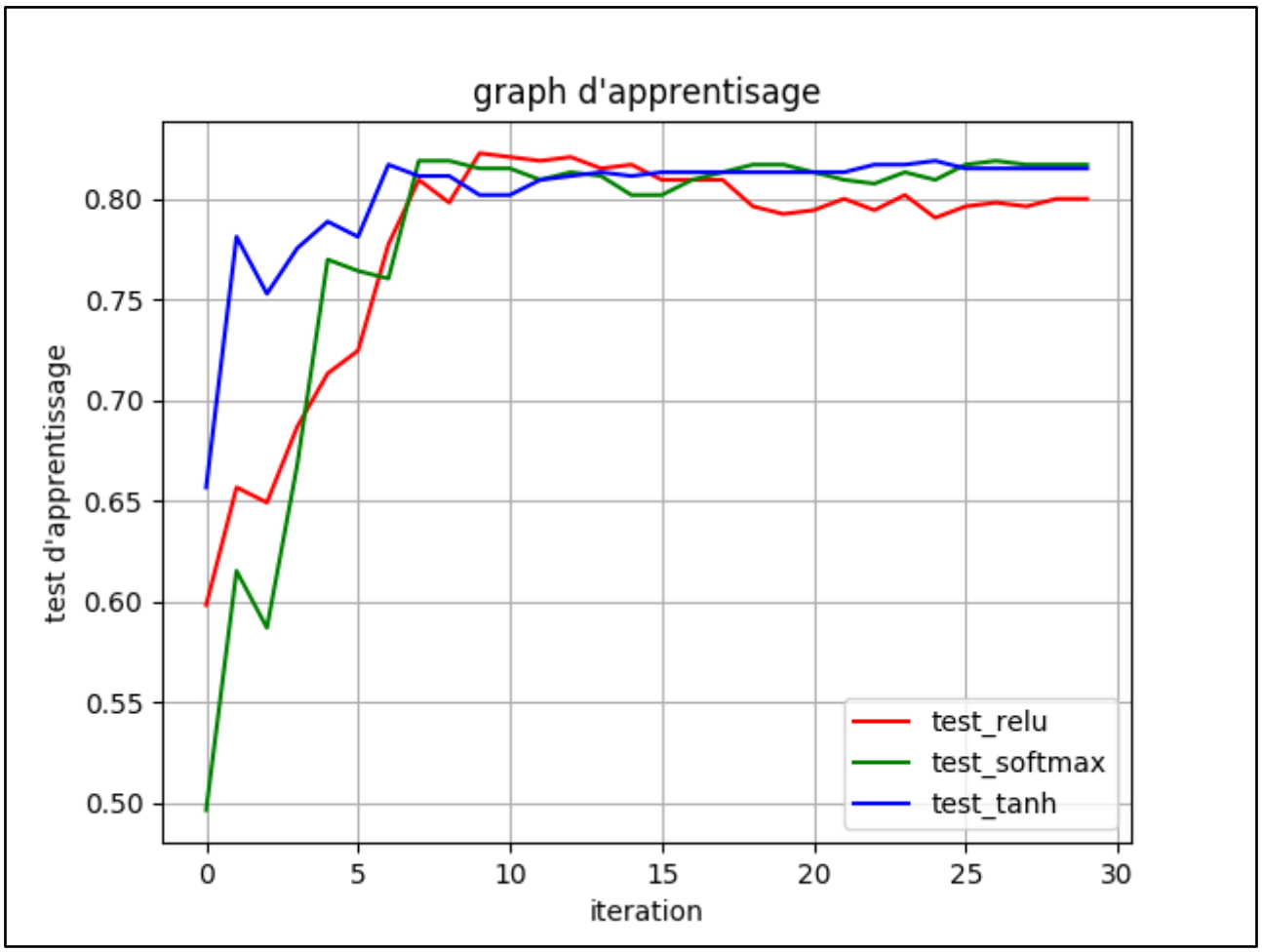

FIGURE 4.2 : GRAPHE EXPLICATIF DU TEST AVEC RELU , SOFTMAX ET TANH.

En remarquant que le test sur un modèle de classification avec une fonction d'activation « Softmax » donne un résultat bon par rapport au test sur les deux modèles qui utilise « ReLu », « Tanh », parce que notre modèle word2vec est construisé par un réseau de neurone de l'architecture CBOW fini par une fonction d'activation « Softmax ». Où le test sur un modèle utilise « ReLu » fixe dans la valeur 80 % et les tests sur un modèle de classification de « Softmax » et « Tanh » donne une valeur supérieure 80.00 % est bonne que Relu.

Le plus important aussi pour compléter cette étude comparative entre les trois modèles de classification c'est le sur-apprentissage de test, le graphe suivant montre la différence entre sur-apprentissage de test avec les fonctions d'activation « Relu, « Softmax » et « Tanh » :

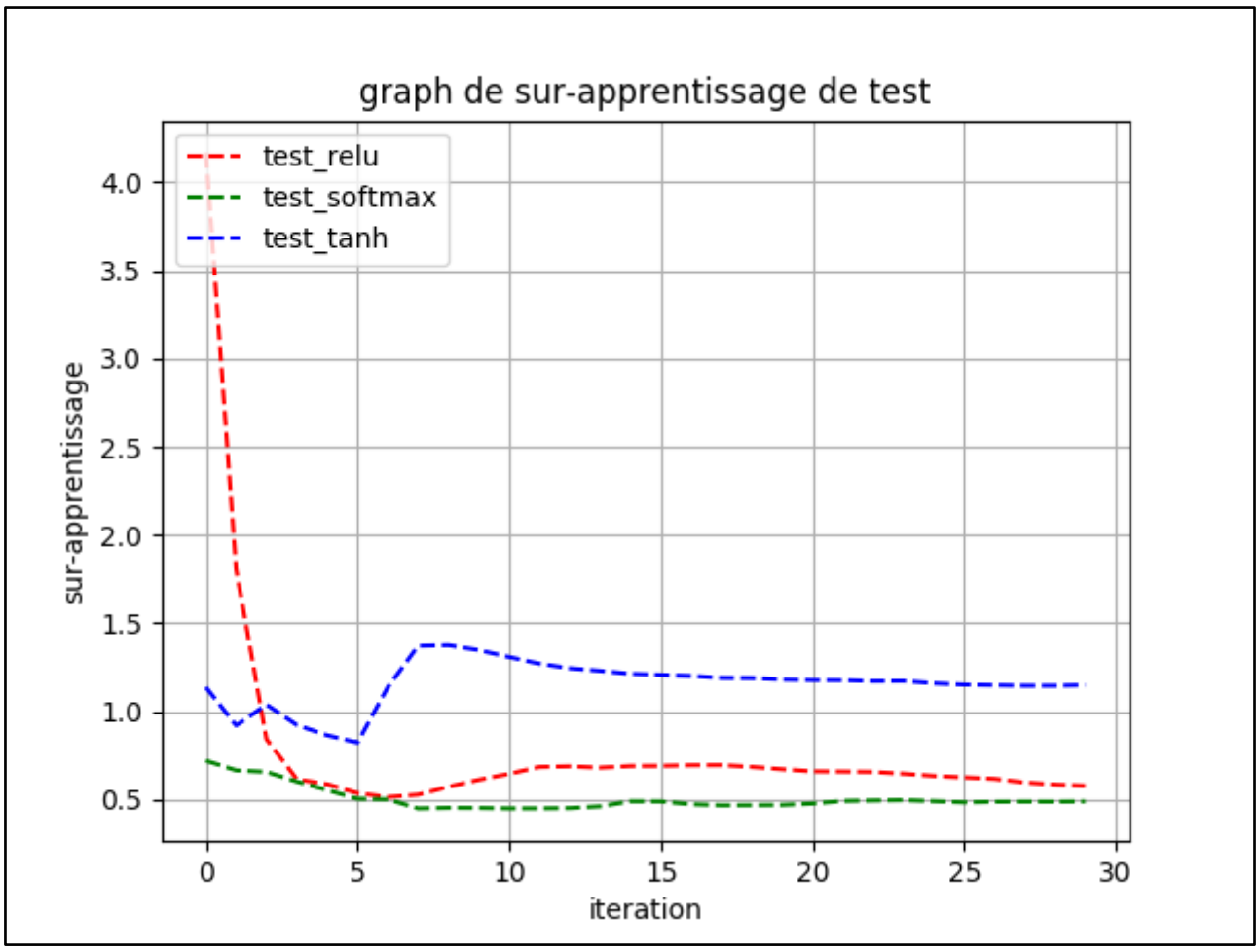

FIGURE 4.3 : GRAPHE DE SUR-APPRENTISSAGE.

En remarquant que le sur-apprentissage de test avec « ReLu », début avec une valeurs élevé de taux d'erreur et démunis avec l'augmentation de nombre d'itération, mais le taux d'erreur de test « Softmax » début avec une valeur inferieur par rapport « ReLu ».

4.3.3. Résultats obtenus par le classifieur CNN

Dans notre test, nous avons mesuré la précision et le rappel de chaque classe dans les trois tests en exploitant la prédiction du test des étapes précédentes, le résultat de chaque test est montré dans les figures 4.4, 4.5 et 4.6.

```
================ evaluation relu ==================
106/106 [==============================] - 4s 42ms/step
================ prediction ====================
Test d'apprentissage: 80.000003
Test sur-apprentissage: 57.782983
                precision    recall  f1-score   support

     1.Buisness       1.00      0.62      0.76        39
       2.Design       0.81      0.69      0.74        42
3.Global Issues       0.91      0.66      0.76        32
      4.Science       0.75      0.57      0.65        37
   5.Technology       0.67      0.73      0.70        56

      micro avg       0.79      0.66      0.72       206
      macro avg       0.83      0.65      0.72       206
   weighted avg       0.81      0.66      0.72       206
    samples avg       0.59      0.56      0.57       206
```

FIGURE 4.4 : RAPPEL ET PRÉCISION DE TEST AVEC RELU.

```
================ evaluation softmax ==================
106/106 [==============================] - 5s 46ms/step
================ prediction ====================
Test d'apprentissage: 81.698112
Test sur-apprentissage: 48.917956
                precision    recall  f1-score   support

     1.Buisness       1.00      0.56      0.72        39
       2.Design       1.00      0.48      0.65        42
3.Global Issues       0.95      0.59      0.73        32
      4.Science       0.95      0.49      0.64        37
   5.Technology       0.78      0.80      0.79        56

      micro avg       0.89      0.60      0.72       206
      macro avg       0.93      0.58      0.71       206
   weighted avg       0.92      0.60      0.71       206
    samples avg       0.55      0.47      0.50       206
```

FIGURE 4.5 : RAPPEL ET PRÉCISION DE TEST AVEC SOFTMAX.

```
================ evaluation tanh ==================
106/106 [==============================] - 4s 41ms/step
================ prediction ====================
Test d'apprentissage: 81.509433
Test sur-apprentissage: 114.896779
                precision    recall  f1-score   support

     1.Buisness       1.00      0.56      0.72        39
       2.Design       0.81      0.62      0.70        42
3.Global Issues       0.90      0.59      0.72        32
      4.Science       0.95      0.49      0.64        37
   5.Technology       0.90      0.64      0.75        56

      micro avg       0.90      0.59      0.71       206
      macro avg       0.91      0.58      0.71       206
   weighted avg       0.91      0.59      0.71       206
    samples avg       0.49      0.47      0.47       206
```

FIGURE 4.6: RAPPEL ET PRÉCISION DE TEST AVEC TANH.

4.3.4. Évaluation de la stratégie d'indexation proposée via le classifieur CNN

Afin d'évaluer les performances de notre stratégie d'indexation sémantique à base de « Word Embedding », nous recourrons à l'utilisation de notre classifier CNN proposé. En effet, dans la section précédente, nous avons prouvé l'efficacité de notre classifieur CNN proposé par les bons résultats obtenus dans la phase des tests. A cet effet, nous utilisons notre classifieur CNN pour une tache de classification des ressources selon les termes d'indexations retenus dans le module d'indexation sémantique. Donc, nous utilisons les indexes retenus pour chaque ressource comme une entrée de tache de classification.

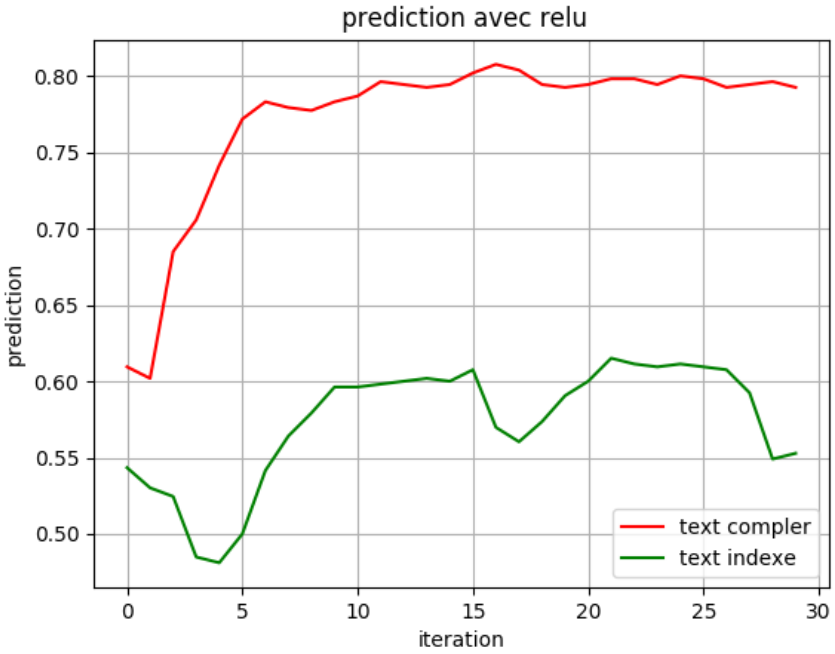

FIGURE 4.7: PRÉDICTION DES TESTS AVEC RELU.

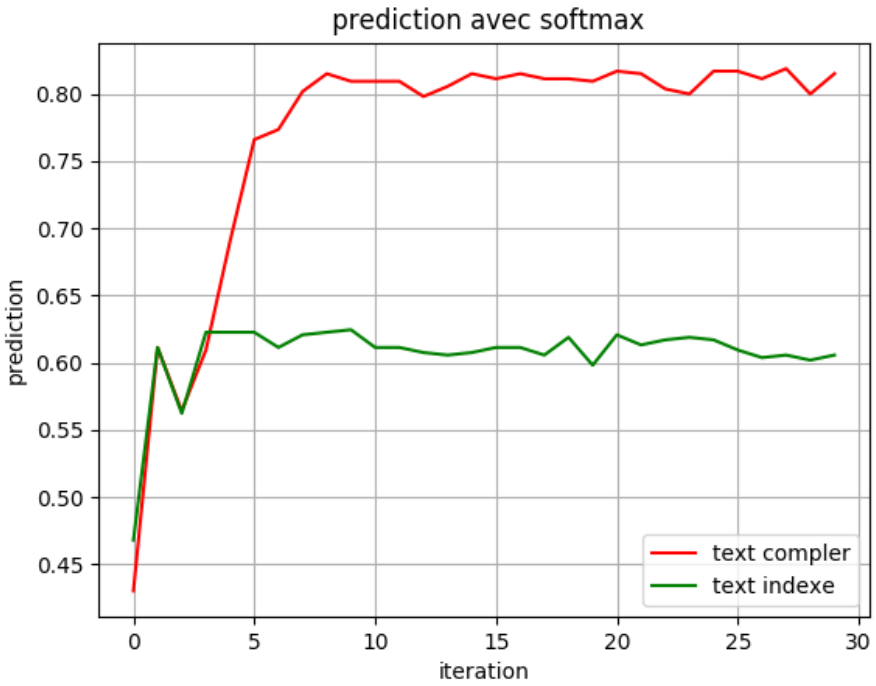

FIGURE 4.8 : PREDICTION DES TESTS AVEC SOFTMAX.

Dans les figures précédentes nous avons présenté la différence entre le test avec les ressources parlées complètes et le test avec les ressources indexées avec notre stratégie d'indexation.

En remarquant que le teste avec les termes d'indexations est fixe dont la valeur supérieur de prédiction égale 62% avec la fonction d'activation « Softmax » mais avec la fonction « Relu » la prédiction égale 61%.

5. Conclusion

Nos expérimentations effectuées sur un corpus «TED» de ressources parlées en anglais de taille moyenne nous ont montré que l'indexation sémantique du contenu par les techniques de « Word *Embedding* » et l'algorithme « Word2Vec » permette de fournir des résultats d'indexation sémantique favorables. En faisant abstraction des temps machine importants dédiés aux opérations d'IS des collections de documents, l'apport de « Word *Embedding* » aux domaine indexation sémantique est sans doute très intéressant et certes, mais comme il exige des ressources lexicales complètes, il est encore tôt de notre point de vue, d'estimer avec certitude le taux de cet apport dans l'amélioration effectives des résultats d'indexation sémantique.

Conclusion Générale

1. Conclusion

Nous avons présenté dans ce manuscrit une étude sur les techniques d'intégration des techniques de « *Word Embedding* » et « Deep Learning » pour le processus d'indexation sémantique des transcriptions automatique du contenu des ressources parlées. Dans ce contexte, nous avons effectué une étude sur les techniques les plus populaires pour l'intégration de mots « Word Embedding » pour le développement d'un support de connaissance pour la tâche d'indexation sémantique.

En plus, ce modèle est utilisé aussi pour diminuer l'impact des erreurs de transcription engendrées par les systèmes de reconnaissance automatique de la parole par l'utilisation des clauses de similarités définies par le modèle « Word2vec ».

Entre autres, nous avons conçu et réalisé un classifier « CNN » avec l'architecture de « Deep Learning » pour évaluer les performances de notre stratégie d'indexation sémantique proposée.

Enfin, pour les tests et l'évaluation de notre approche, nous avons utilisé un extrait du corpus de la communauté « *TED Talks* ». Le choix de ce corpus est fondé par sa disponibilité contrairement aux autres ressources parlées.

2. Perspectives futures

Pour nos perspectives futures, nous visons d'accentuer nos recherches sur les aspects suivants :

- L'enrichissement du corpus utilisé pour la création du modèle
- La proposition d'une stratégie de détection des seuils optimaux pour les bornes de l'intervalle sémantique utilisé dans la sélection des termes d'indexation.
- L'intégration d'une stratégie d'enrichissement sémantique pour les termes d'indexation détectée.

Références bibliographiques

Références bibliographiques

A

[Abrahams S & al, 2016]	S. Abrahams, H. Danijar, E. Erik, et S. Ariel. n.d. *Tensorflow For Machine Intelligence*.2016
[Aizenberg I & al, 2013]	I. Aizenberg, N. N. Aizenberg et J. P. Vandewalle, «Multi-Valued and Universal Binary Neurons :Theory, Learning and Applications,» Springer Science & Business Media, 2013 .
[Agirre E & Edmonds P.G, 2006]	E. Agirre, P.G. Edmonds, Word Sense Disambiguation: Algorithms and Applications. Springer; 2006.
[Andrei M.B & al, 2017]	M.B. Andrei, T Radu, C. Iones, From Image Image to Text Classification:A Novel Novel Approach based on Clustering Clustering Word Embeddings 2017.
[Azpiazu I.M & al, 2018]	I.M. Azpiazu, D. Nevena, A. Oghenemaro, et S.P. Maria, Looking for the Movie Seven or Sven from the Movie Frozen : A Multiperspective Strategy for Recommending Queries for Children. In Proceedings of the 2018 Conference on Human Information Interaction& Retrieval. ACM, 92–101, 2018.

B

[Bay H & al, 2008]	H. Bay, A. Ess,T. Tuytelaars, L.V. Gool, Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding , 110(3), 346–359 , 2008.
[Baziz M, 2005].	M. Baziz, Indexation conceptuelle guidée par ontologie pour la recherche d'information. Thèse de doctorat, Université Paul Sabatier, 2005.
[Bendib I, 2018]	I. Bendib La recherche dans le contenu Parlé. Thèse de Doctorat, Université Badji Mokhtar Annaba- Algerie, 2018
[Bourez C, 2017]	C. Bourez, *Deep Learning With Theano*. Birmingham : Packt Publishing, 2017.

D

[Dai A.M & al, 2015]	A.M. Dai, C. Olah et Q.V. Le Document Embedding with Paragraph Vectors,2015.
[Dechter R & al, 1986]	R. Dechter et J. Pearl, « The cycle-cutset method for improving search performance in AI applications. University of California, Computer Science Department, 1986.».
[Deerwester S & al, 1990]	S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391–407, 1990.
[Deng L & Yu D, 2014]	L. Deng, et D. Yu, *Deep learning*. Boston, p.217.2014.
[Dima S & Arafat S, 2018]	D. Suleiman,et A. Awajan, Comparative study of word embeddings models and their usage in Arabic language applications, 2018.
[Dumais S.T, 2004].	S. T. Dumais, "Latent semantic analysis", Annual review of informationscience and technology, vol. 38, pp. 188-230, 2004

F

[Fankam C & al, 2009]	C. Fankam, L. Bellatreche, D. Hondjack, Y. A. Ameur and G. Pierra, Conception de Bases de Données à partir d'Ontologies de Domaine, Technique et Science Informatiques, p. 1233–1261, 2009.

G

[Gomez P, 1999]	A. Gomez Perez, Ontological Engineering : A State of the Art, Expert Update 2, , pp. 33-44, 1999.
[Gurber T, 1993]	T. Gruber, "A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition," Current issues in knowledge modeling, pp. 199-220, 1993.
[Guarino N, 1995]	N. Guarino, Formal Ontology, Conceptual Analysis and Knowledge Representation, International Journal of Human-Computer Studies Volume 43, Issues 5–6, p. 625–640., 1995.
[Guarino N & al, 1999]	N. Guarino N., C. Masolo, and G. Vetere, OntoSeek : Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs. National Research Council, LADSEBCNR : Padavo, Italy, 1999
[Gysel R & al, 2018]	R. Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2018. Neuralvector spaces for unsupervised information retrieval. ACM Transactions on Information Systems (TOIS) 36, 4 , 38,2018.

H

[Hubert G, 2009]	G. Hubert, J. Mothe and B. Ralalason, Modèle d'indexation dynamique à base d'ontologies, In Conférence en Recherche d'Information et Application., 2009.

I

[Ionescu R.T & al, 2016]	R.T. Ionescu, M. Popescu, Knowledge Transfer between Computer Vision and Text Mining. Advances in Computer Vision and Pattern Recognition. Springer International Publishing; 2016.

J

[Jean C & al, 2018]	C. jean et W. Christian. Using Word Embeddings for Unsupervised Acronym Disambiguation.2018.
[Josef S & al, 2004]	S. Josef et K. Jezek.. "Using latent semantic analysis in text summarization and summary evaluation." Proc. ISIM',2004
[Ju R & al , 2015]	R. Ju, P. Zhou, C.H. Li et L. Liu, An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis. 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing.2015.

K

[Kamal A & al, 2018]	A. Kamal & Z. Zhang & K. Yang. Latent Semantic Analysis Approach for Document Summarization Based on Word Embeddings. 2018.
[Karbasi, 2007]	
[Koopman R & al, 2015]	R. Koopman, S. Wang, et A. Scharnhorst. Contextualization of Topics - Browsing through Terms, Authors, Journals and Cluster Allocations. Proceedings of ISSI 2015 Istanbul. 15th International Society of Scientometrics and Informetrics Conference, Istanbul, Turkey, 29th June to 4th July 2015, pages 1042–1053, 2015.
[Koopman R & al, 2017]	R. Koopman, S. Wang, et A. Scharnhorst. Contextualization of topics – browsing through the universe of bibliographic information. In J. Gläser, A. Scharnhorst, and W. Glänzel, editors, Same data – different results? Towards a comparative approach to the identification of thematic structures in science, Special Issue of Scientometrics. 2017.
[Kowalski G, 2010]	G. Kowalski, Information Retrieval Architecture and Algorithms, Springer-Verlag New York, Inc., New ,York, NY, 2010

L

[Lowe D.G, 1999]	D.G. Lowe, Object Recognition from Local Scale-Invariant Features. Proceedings of ICCV 1999, 2, 1150–1157,1999.

[Lowe D.G, 2004]	D.G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 2004, 60(2), 91–110. 2004.
[Luhn H.P, 1959]	H.P Luhn. Keyword-in-Context Index for Technical Literature (KWIC Index). International Business Machines Corp. Yorktown Heights, NY, 1959

M

[Mahdaouy A & al, 2016]	A. Mahdaouy, E. Gaussier, and S. Ouatik El Alaoui, Arabic Text Classification Based on Word and Document Embeddings. International Conference on Advanced Intelligent Systems and Informatics, 2016.
[Matt J.K & al, 2015]	J.K. Matt, S. Yu, I.K. Nicholas, et Q.W. Kilian, From WordEmbeddings To Document Distances. Proceedings of The 32nd International Conference on Machine Learning, 37:957–966, 2015.
[McClure N, 2017]	N. McClure, *Tensorflow Machine Learning Cookbook*. S.L: Packt Publishing.2017
[Michael F & al, 2015]	F. Michael & V. Dmitry et K. Pushmeet, PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions. 2015.
[Mikolov T & al, 2013a]	T. Mikolov ,I. Sutskever,K. Chen, G. Corrago, J. Dean, Distributed representations of words and phrases and their compositionality. In: 27th Conference on Neural Information Processing Systems; 5-10 December 2013; Lake Tahoe, Nevada, USA. pp. 1-9.2013.
[Mikolov T & al, 2013b]	T. Mikolov , K. Chen, G. Corrago, J. Dean. Efficient estimation of word representations in vector space. In: Interna-tional Conference on Learning Representations; Scottsdale, Arizona, USA. pp. 1-12. 2-4 May 2013.
[Moolayil J & al, 2019]	J. Moolayil, et S. John. *Learn Keras For Deep Neural Networks*. [S.l.]: Apress.2019.

P

[Philbin J & al, 2007]	J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching. Proceedings of CVPR ,1-8,2007.
[Pennington J, 2014]	J. Pennington et R. Socher, Manning CD (2014) Glove: global vec-tors for word representation. In: Empirical methods in natural lan-guage processing (EMNLP), 2014. Association for Computational Linguistics, Doha, pp 1532–1543,2014.
[Porter M.F, 1980]	M.F. Porter, An algorithm for suffix stripping Program, Vols. 14-3, Program, 1980.

Q

[Qv Le & Mikolov T, 2014]	Qv Le and T. Mikolov. Distributed Representations of Sentences and Docu-ments. International Conference on Machine Learning - ICML 2014, 32:1188–1196,5 2014.

R

[Rajaraman A & al, 2011]	A. Rajaraman., J.D. Ullman, "Data Mining". Mining of Massive Datasets. pp. 1–17,2011.

S

[Salton G & al, 1975]	G. Salton, A. Wong, et C.S. Yang, "A Vector Space Model for Automatic Indexing", Communications of the ACM, vol. 18, pp. 613-620, 1975.
[Salton G & al, 1993]	G. Salton and M. J. McGill. Introduction to Moderne Information Retrieval, New York, 1983.
[Santos D.C & al, 2014]	D.C. Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical

	Papers. 69–78,2014.
[Sean B, 2003]	B. Sean, . M. Ralf and C. Peter, he {DIG} Description Logic Interface, in Proceedings of the 2003 International Workshop on Description Logics, Rome Italy, 2003.
[Sy M.F, 2012]	M.-F. Sy, Utilisation d'ontologies comme support à la recherche et à la navigation dans une collection de documents, Thèse de Doctorat, Université Montpellier II, 2012.

Y

[Yuhua L. & al, 2006]	L. Yuhua, D. Mclean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Transactions on Knowledge and Data Engineering*, *vol. 18*, pp. 1138 – 1150, 2006.
[Yin W & al, 2017]	W. Yin, K. Kann,M. Yu, et H. Schütze. Comparative Study of CNN and RNN for Natural Language Processing, 2017.
[Young T& al, 2018]	T. Young, D. Hazarika, S. Poria et E. Cambria.Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine 13, 3 (2018), 55–75, 2018.

Z

[Zhang, J & al, 2007]	J. Zhang, M. Marszalek,S. Lazebnik et C. Schmid, Local Features and Kernels for Classification of Texture and Object Categories: AComprehensive Study. International Journal of Computer Vision 2007, 73(2), 213–238.2007.