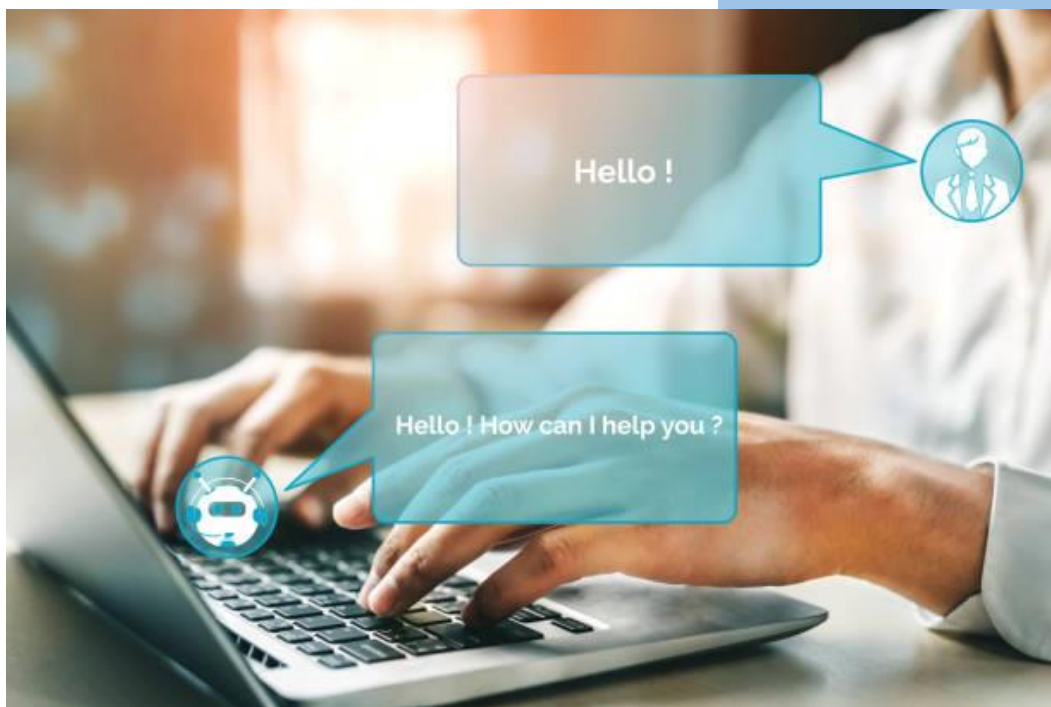


## PROJET DE RECHERCHE TECHNOLOGIQUE

# Chatbot pour plateforme LMS et Rainbow Classroom



### Encadrants

Ahmed SAMET

François DEBEUVRON

### Etudiants

Achta NGOMA

Nouhaila EL ALAOU

## Sommaire

Introduction .....	4
1. Présentation du sujet.....	4
2. Etat de l’art .....	5
3. Etudes bibliographiques .....	7
3.1 Natural Language Processing (NLP) .....	8
3.2 Lemmatisation.....	9
3.3 Tokenisation.....	9
3.4 Détermination de la similarité.....	9
3.5 Stop words .....	10
3.6 Word Embedding .....	10
3.7 Question-answering .....	14
4. Processus de mise en place du chatbot.....	17
4.1 Extraction du texte dans des fichiers.....	17
4.2 Tests de quelques méthodes d’embedding.....	18
4.3 Question-réponse avec cdQA sur un ensemble de fichiers.....	21
4.4 Haystack.....	23
Conclusion.....	32

## Table des illustrations

Figure 1 : Agenda du projet.....	5
Figure 2 : Etapes de fonctionnement du bot V1 .....	8
Figure 3 : Etapes de fonctionnement du bot V2.....	8
Figure 4 : Quelques méthodes d'embedding .....	11
Figure 5 : Modèle CBOW .....	15
Figure 6 : Modèle Skip Gram.....	15
Figure 7 : Comparaison entre CBOW et Skip-gram .....	16
Figure 8 : Pré-entraînement BERT.....	17
Figure 9: le code de Bert-multilingual .....	21
Figure 10: Bert_multilingual pour le Tokenizer .....	21
Figure 11: Résultat de test.....	21
Figure 12 : Architecture cdQA.....	22
Figure 13 : Architecture Haystack .....	24
Figure 14: Installation de Haystack.....	25
Figure 15: Initialisation du Retriever .....	25
Figure 16: Haystack version anglaise.....	26
Figure 17: Haystack version française .....	26
Figure 18: Création du Pipeline .....	26
Figure 19: Commande de la prédiction .....	26
Figure 20: Résultat du code.....	26
Figure 21 : QA avec mots vides .....	28
Figure 22 : QA sans mots vides .....	29

# Introduction

Un chatbot aussi appelé dialogueur ou agent conversationnel est un agent logiciel qui dialogue avec un utilisateur. C'est une machine créée pour converser avec une personne en donnant l'impression de discuter avec une autre personne.

La recherche sur cette interface personne-machine a été influencée par la compétition sur le test de Turing (1950) qui consiste à donner l'illusion qu'un programme pense par un dialogue sensé. Autrement dit, un utilisateur humain formule une demande en langage naturel et le chatbot traite la demande. De nos jours, les chatbots ne se limitent plus à des questions de base, mais intègrent désormais des algorithmes plus évolués permettant une gestion des échanges d'un niveau de complexité plus élevé qu'auparavant. Ils fleurissent sur internet et les entreprises en profitent pour améliorer leur service et traiter les demandes des clients de manières efficaces. Aussi, des chatbots éducatifs se développent de plus en plus pour améliorer l'apprentissage. C'est dans cette optique que se classe notre projet de recherche technologiques (PRT).

## 1. Présentation du sujet

Rainbow Classroom d'Alcatel-Lucent est une plateforme intuitive pour dispenser des cours en ligne. Avec la crise COVID, les établissements ont compris l'importance de mettre en place des cours virtuels afin d'assurer la continuité pédagogique. Rainbow Classroom est une solution qui intègre des fonctionnalités de communication et de collaboration en temps réel pour les différentes parties prenantes.

Dans le but de parfaire la plateforme, nous travaillons en collaboration avec l'entreprise Alcatel Lucent Entreprise et des étudiants en informatique du CESI sur le développement d'un chatbot pour plateforme LMS (exemple Moodle) et Rainbow Classroom en utilisant l'intelligence artificielle. Cela permettra :

- Aux professeurs de fournir des détails sur les cours sont pour autant répondre à des questions répétitives
- Aux élèves, d'avoir une réponse adaptée à leur question rapidement

Les chatbots ne remplacent pas l'enseignant mais le secondent et le complètent. Cet outil est un véritable compagnon pour l'élève et permet de différencier les apprentissages pour chacun car la conversation s'adapte aux réponses et aux demandes de celui-ci.

Dans ce travail collaboratif entre les différentes parties, nous devons effectuer une étude pour déterminer les méthodes à utiliser telle sorte à être un gagne temps dans l'étape de mise en place pour les étudiants du CESI. Les **objectifs de notre étude** sont les suivants :

- Extraction du texte dans les différents formats de fichiers
- Déterminer les meilleurs méthodes d'embedding

- Transcription d'un fichier audio ou vidéo
- Recherche d'informations dans un très grand nombre de documents

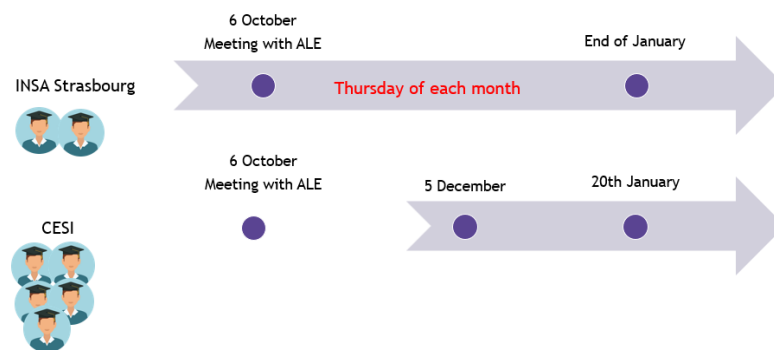


Figure 1 : Agenda du projet

## 2. Etat de l'art

Cette application est vraiment courante dans le secteur éducatif. Ils existent différents travaux qui ont été réalisés sur le développement de chatbot pour améliorer les conditions d'apprentissage des étudiants en leur permettant de facilement retrouver la réponse à leur question et automatiser une partie du travail des enseignants. Les articles et ouvrages suivants présentent quelques applications ainsi que les connaissances à avoir pour les mettre en œuvre.

### **Article 1 :** Chatbot de réponse aux questions à l'aide du Deep Learning avec NLP

Cet article de quatre chercheurs indiens en ingénierie et informatique à Institut national de technologie, Tiruchirappalli, Tamil Nadu, Inde et publié en juillet 2021 traite du développement d'un "Question-Answering Chatbot" pour le domaine de l'éducation afin d'aider les étudiants à répondre à leurs questions sur les sujets et les cours et de soulager les enseignants. Pour ce faire, leur travail a été divisé en trois grandes étapes.

La première consiste à créer une base de connaissances qui exploite la spécialité du domaine spécifique. Ceux sont les étudiants qui intègrent les fichiers PDF dans lesquels ils cherchent des informations. Ils subissent plusieurs transformations NLP telles que la tokenisation, la lemmatisation, la singularisation, la suppression de la ponctuation ... afin de réduire la taille du vocabulaire dans la base de connaissances finale. Enfin, le texte est enregistré en tant que base de connaissances.

Dans la deuxième étape, le chatbot utilise les méthodes de récupération d'informations pour vérifier la similitude entre la question et les phrases dans la base de connaissances.

La dernière étape nécessite de reclasser les documents sélectionnés, ceci est possible en utilisant des réseaux de neurones convolutifs (CNN).

La performance du chatbot a été mesurée au travers de deux grandeurs: la précision moyenne (MAP) et le rang réciproque moyen (MRR). La précision moyenne calcule les rangs de toutes les réponses correctes, et le rang réciproque moyen calcule le rang de la première réponse correcte. A la fin des tests, ces grandeurs s'élèvent à 77,42% pour MAP et 84,51% MEP. Ce qui est plutôt correcte comparé aux autres travaux sur le marché.

[Chatbot de réponse aux questions à l'aide du Deep Learning avec NLP | Publication de la conférence IEEE | IEEE Xplore \(insa-strasbourg.fr\)](#)

[Question-Answering-chatbot-for-education/components.py at main · NamrataKadam/Question-Answering-chatbot-for-education \(github.com\)](#)

## **Article 2 : Un système de réponses aux questions pour les étudiants**

Cette étude réalisée par trois chercheurs du département d'information et d'informatique, Chung Yuan Christian University, Taoyuan, Taïwan a été menée en collaboration avec 67 étudiants dans un cours de communication en Soft Power.

Les travaux ont débuté par le traitement de données (soixante-deux fichiers vidéo de légendes et soixante-seize courts articles en cinq chapitres). Ces données sont traitées pour former une base de données et la particularité de cette étude est l'emploi d'un modèle LDA pour classifier les données selon un sujet précis.

La dernière étape : question answering , a été divisé en trois modules, analyse de phrases, récupération de données et recherche de réponses. En fonction de la question posée, le bot affiche la liste des phrases similaires et en option, le nom de la vidéo ou du document ainsi que le début de la phrase.

[Chatbot : un système de réponse aux questions pour les | étudiants Publication de la conférence IEEE | IEEE Xplore \(insa-strasbourg.fr\)](#)

## **Resource 1 : Modèle BERT**

Dans les systèmes actuels de réponse aux questions, les méthodes de similarité sont divisés en deux catégories : es méthodes d'apprentissage profond et les méthodes conventionnelles. Les méthodes conventionnelles reposent fortement sur des caractéristiques artificielles, ont une

faible capacité de généralisation et une précision insuffisante. RNN et CNN ont également une extraction globale de fonctionnalités textuelles. Cet article propose un modèle hybride de correspondance des réponses aux questions basé sur BERT, qui utilise le modèle de préformation BERT pour capturer et représenter l'information sémantique de la phrase QAS et la pertinence sémantique entre les deux. Le vecteur caractéristique généré par le modèle BERT est utilisé comme Bi-LSTM \_ GCN pour l'entrée du modèle, l'extraction de caractéristiques est effectuée pour obtenir davantage les caractéristiques syntaxiques de la phrase, et enfin le mécanisme d'attention est ajouté pour trouver la réponse cible, et l'efficacité de l'algorithme proposé est vérifiée sur les deux types d'ensembles de données.

[Modèle d'appariement de réponses mixtes aux questions BERT, | Publication de la conférence IEEE | IEEE Xplore \(insa-strasbourg.fr\)](#)

### **Ressource 2 : Ouvrage sur le NLP**

La conception de notre chatbot requiert un grand nombre de connaissances dans le domaine du NLP. Cet ouvrage des éditions PACKT intitulé “ Hands-on Natural language processing with python” est un guide pertinent pour la compréhension et la mise en œuvre de différentes étapes du projet. Il présente différentes applications du NLP en lien avec notre projet tels que : la transcription d'une vidéo en texte et les chatbots de question-answering.

[Hands-On Natural Language Processing with Python - ScholarVox Université \(insa-strasbourg.fr\)](#)

## **3. Etudes bibliographiques**

La connaissance bibliographique du sujet est primordial pour mieux comprendre le sujet. D'autant plus qu'il existe plusieurs technologies et méthodes pour la création d'un chatbot. Nous souhaitons que le bot fonctionne ainsi : le professeur crée un ensemble d'intentions qui correspondent aux différentes questions que les étudiants pourraient demander. Si le bot n'a pas trouvé de réponse dans la base de données de cours, il renvoie des recommandations google, YouTube ...

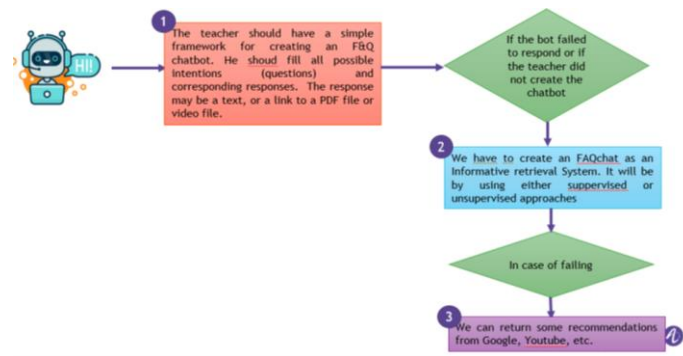


Figure 2 : Etapes de fonctionnement du bot V1

Toutefois, avec une meilleure analyse du sujet, nous avons modifié le fonctionnement pour plus de praticité. En effet, écrire l'ensemble de questions que pourraient avoir les élèves peut facilement devenir pénible pour l'enseignant, le chatbot perdrait alors son utilité. Pour y palier, nous nous sommes orientés vers un apprentissage non supervisé. A partir d'un ensemble de données disponible sur la plateforme LMS ou Rainbow Classroom, le bot doit tous les examiner pour trouver la bonne réponse. Dans un premier temps, nous commençant par un chatbot par cours.

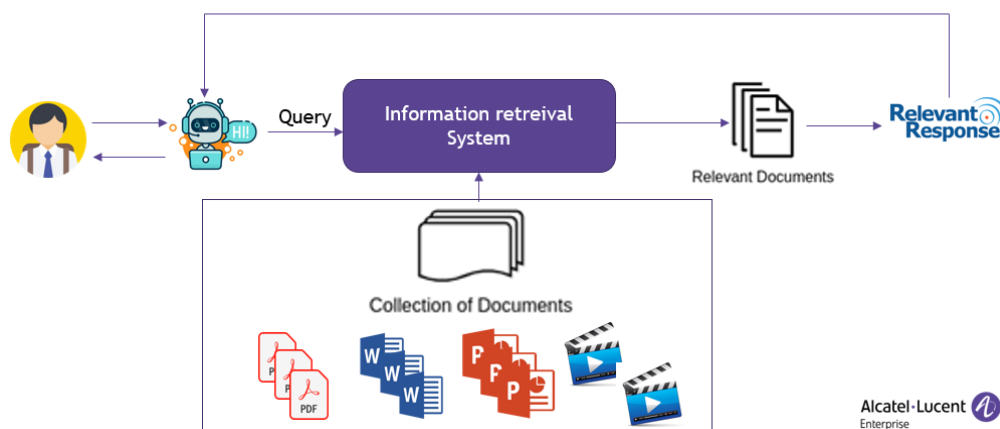


Figure 3 : Etapes de fonctionnement du bot V2

Le contexte ayant été introduit, voici une liste d'éléments nous permettons de le mettre en œuvre.

### 3.1 Natural Language Processing (NLP)

Traitement automatique du langage naturel est une suite de programmes informatiques développés dans le but de comprendre le langage tel qu'il est écrit ou parlé par les humains. Il



résout un certain nombre de tâches comme la traduction automatique, la vérification orthographique ou grammaticale, la classification de texte, la reconnaissance d'entités nommées, leur sélection, la production de résumé, à travers une succession des étapes suivantes :

1. Saisir le texte (ou le son ou la vidéo converti en texte)
2. Segmentation du texte en composantes (segmentation et tokenisation).
3. Nettoyage du texte (filtrage de la « corbeille ») aussi suppression des éléments inutiles.
4. Vectorisation du texte et ingénierie des éléments.
5. Lemmatisation et racinisation – réduction des inflexions pour les mots.
6. Utilisation d'algorithmes et de méthodes d'apprentissage automatique pour les modèles d'entraînement.
7. Interprétation du résultat.

### 3.2 Lemmatisation

Cette méthode va au-delà de la réduction des mots et considère le vocabulaire complet d'une langue pour appliquer une analyse morphologique aux mots, visant à supprimer uniquement les terminaisons flexionnelles et à renvoyer la forme de base ou de dictionnaire d'un mot, connue sous le nom de lemme, c'est contrairement au stemming.

### 3.3 Tokenisation

En informatique, l'analyse lexicale ou tokenisation est le processus de conversion d'une séquence de caractères tel qu'un texte en une séquence de jetons lexicaux (chaînes avec une signification assignée et donc identifiée).

### 3.4 Détermination de la similarité

L'évaluation de la similarité entre les documents textuels est l'un des sujets importants dans plusieurs domaines tels que l'analyse de données textuelles, la recherche d'informations et l'extraction de connaissances à partir de données textuelles (Text Mining). Dans chacun de ces domaines, les similarités sont utilisées pour différents traitements, premièrement, en analyse de données textuelles, les similarités sont utilisées pour la description et l'exploration de données, ensuite en recherche d'information, l'évaluation des similarités entre documents et requêtes est utilisée pour identifier les documents pertinents par rapport à des besoins d'information exprimés par les utilisateurs, et en Text Mining, les similarités sont utilisées pour produire des représentations synthétiques de vastes collections de documents.

Il existe deux types de similarité : Similarité syntaxique permet de comparer des documents textuels en se basant sur les chaînes de caractères qui les composent en passant par la représentation vectorielle. Par exemple, les chaînes de caractères "voiture" et "voiturier" peuvent être considérées comme très proches, alors que "voiture" et "automobile" pourront être considérées comme très différentes. Et le deuxième type est la similarité sémantique est un concept selon lequel un ensemble de documents ou de termes se voient attribuer une métrique basée sur la ressemblance de leur signification / contenu sémantique.

### 3.4.1 *Les méthodes de mesure de la similarité*

**Similarité Jaccard (similarité syntaxique)** : C'est la forme la plus basique de calcul de similarité entre deux textes car elle ne nécessite pas d'avoir une représentation numérique des textes. L'indice de Jaccard est mesuré en divisant le nombre de mots partagés par les deux textes par le nombre total de mots.

**Distance euclidienne** : c'est la distance la plus courte entre deux vecteurs représentant les documents.

**Similarité cosinus** : la similarité cosinus est fréquemment utilisée en tant que mesure de ressemblance entre deux documents. Il pourra s'agir de comparer les textes issus d'un corpus dans une optique de recherche d'information. C'est-à-dire, un document vectorisé est constitué par les mots de la requête et est comparé par mesure de cosinus de l'angle avec des vecteurs correspondant à tous les documents présents dans le corpus avec aussi et on évalue ainsi lesquels sont les plus proches.

## 3.5 Stop words

Stop words ou mot vide désigne tous les mots n'ayant pas de réelle signification comme les articles, et, mais et comment... En effet, ils sont si courants et reviennent de façon tellement régulière qu'ils ne permettent pas de caractériser, au sens lexical, un texte par rapport à un autre texte. Par exemple, si nous souhaitons rechercher "Qu'est-ce qu'une carte mère ?" sur Google, le moteur de recherche ne recherche que le terme « carte mère ».

## 3.6 Word Embedding

C'est un ensemble de techniques de machine Learning qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels, décrits dans un modèle vectoriel. Ces

nouvelles représentations de données textuelles ont permis d'améliorer les performances des méthodes de traitement automatique des langues. C'est une étape essentielle car nos données sont essentiellement des textes mais l'ordinateur ne comprend que des nombres. Il existe plusieurs méthodes de Word Embedding dépendantes au indépendantes du contexte.

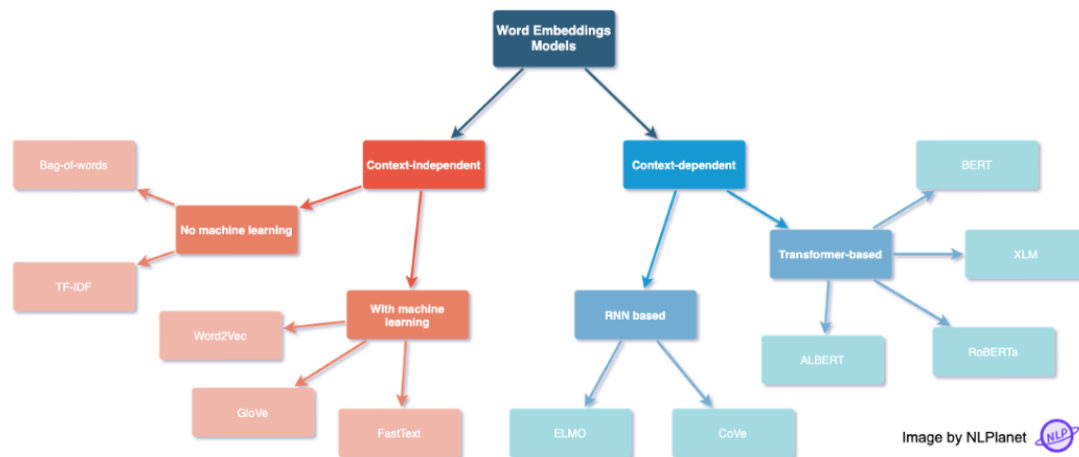


Figure 4 : Quelques méthodes d'embedding

### 3.6.1 Indépendant du contexte sans machine Learning

**Bag of Words** est la technique qui était majoritairement utilisée avant l'apparition des modèles basés sur les réseaux de neurones. Ce modèle se préoccupe seulement de savoir si des mots connus se trouvent dans le document et non pas où ils se trouvent. Elle élimine toute l'information relative à l'ordre des mots et met l'accent sur l'occurrence des mots dans un document. En effet, le bag-of-words est :

- Très simple à comprendre,
- Peut être utilisé sur un faible échantillon de données alors que les techniques basées sur les réseaux de neurones ont en besoin d'énormément.
- Ne nécessite pas de capacités de calcul importantes (un CPU suffit alors que les techniques de réseaux de neurones nécessitent des GPUs ou des TPUs qui coûtent extrêmement chers).

**TF-IDF** : il s'agit d'une des techniques de traitement du langage naturel les plus populaires et les plus efficaces, il permet d'estimer l'importance d'un mot par rapport à tous les autres termes dans un texte ou dans une collection de textes. Cette technique est divisée en deux parties : TF, Term Frequency qui s'occupe à calculer le nombre d'occurrences d'un terme en comparaison avec le nombre total de mots dans un texte. IDF:Inverse document Frequency, mesure l'importance d'un terme dans un texte ou une collection de texte. Il est calculé comme un logarithme du nombre de documents divisé par le nombre de documents contenant ce terme. Alors, l'utilisation de la technique TF-IDF revient à évaluer les valeurs TF pour chaque mot, extraire les valeurs IDF pour ces termes pour enfin obtenir les quotients TF-IDF en multipliant les résultats de TF par IDF. Cela forme un dictionnaire de classification pour chaque mot utilisé, suivant leur importance.

### 3.6.2 *Indépendant du contexte avec machine learning*

**Word2Vec** repose sur des réseaux de neurones à deux couches et cherche à apprendre les représentations vectorielles des mots composant un texte, de telle sorte que les mots qui partagent des contextes similaires soient représentés par des vecteurs numériques proches. Cet algorithme possède deux architectures neuronales, CBOW et Skip-Gram, parmi lesquelles l'utilisateur peut choisir. CBOW reçoit en entrée le contexte d'un mot, c'est-à-dire les termes qui l'entourent dans une phrase, et essaye de prédire le mot en question. Or Skip-Gram fait exactement le contraire : elle prend en entrée un mot et essaye de prédire son contexte. Dans les deux cas, l'entraînement du réseau se fait en parcourant le texte fourni et en modifiant les poids neuronaux afin de réduire l'erreur de prédiction de l'algorithme. Pour les applications réelles, les Word2Vec sont créés à l'aide de milliards de documents, par exemple pour l'application Google est formé à l'aide de 3 millions mots et phrases.

**Glove**: Algorithme d'apprentissage non supervisée permettant d'obtenir des représentations vectorielles de mots. Son entraînement est basé sur des statistiques de cooccurrences de mots globales et agrégées calculées sur de vastes corpus.

**FastText**: Cet algorithme peut accomplir une grande exécution pour la représentation des mots et le regroupement des phrases, uniquement en raison de mots peu communs en utilisant des données au niveau des caractères.

### 3.6.3 *Dépendant du contexte et basé sur RNN*

**ELMO (Embeddings from Language Model)** : algorithme développé en 2018 par AllenNLP, il apprend les représentations de mots contextualisées basées sur un modèle de langage neuronal avec une couche de codage basée sur les caractères et deux couches BiLSTM. Contrairement à GloVe, Bag of Words et Word2Vec les intégrations ELMo sont capables de capturer le contexte du mot utilisé dans la phrase et peuvent générer différentes intégrations pour le même mot utilisé dans un contexte différent dans différentes phrases. Ce modèle a atteint des performances de pointe sur de nombreuses tâches populaires, notamment la réponse aux questions, l'analyse des sentiments et l'extraction d'entités nommées.

**CoVe (Contextualized Word Vectors)** : utilise un encodeur LSTM profond à partir d'un modèle attentionnel de séquence à séquence formé à la traduction automatique pour contextualiser les vecteurs de mots.

### *3.6.4 Dépendant du contexte et basé sur un transformateur*

**BERT (Bidirectional Encoder Representations from Transformers)** : modèle de langage développé par Google en 2018. Cette méthode a permis d'améliorer significativement les performances en traitement automatique des langues. Il est basé sur un transformateur entraîné sur un vaste corpus inter-domaines. Applique un modèle de langage masqué pour prédire les mots qui sont masqués de manière aléatoire dans une séquence, suivi d'une tâche de prédiction de phrase suivante pour apprendre les associations entre les phrases.

Contrairement aux précédentes méthodes séquentielles (word2vec et autres) qui lisent les phrases dans un seul sens et qui utilisent un seul vecteur par mot, le fonctionnement bidirectionnel de BERT permet de lire les phrases dans un sens (de gauche à droite) et dans l'autre (de droite à gauche). Cela permet une vraie amélioration pour la compréhension des requêtes longues avec un système de probabilité d'apparition d'un mot dans un contexte. Ainsi un même mot peut à présent avoir plusieurs vecteurs selon son contexte.

**XLM (Cross-lingual Language Model)** : il s'agit d'un transformateur préentraîné à l'aide de la prédiction du jeton suivant, d'un objectif de modélisation du langage masqué de type BERT et d'un objectif de traduction. XLM utilise une technique de prétraitement connue (BPE) et un mécanisme de formation bilingue avec BERT afin d'apprendre les relations entre les mots dans différentes langues. Le modèle surpasse les autres modèles dans une tâche de classification multilingue (implication de phrases en 15 langues) et améliore considérablement la traduction automatique lorsqu'un modèle pré-entraîné est utilisé pour l'initialisation du modèle de traduction.

**RoBERTa (Robustly Optimized BERT Pretraining Approach)** : il s'appuie sur BERT et modifie les hyperparamètres clés, supprimant l'objectif de préformation de la phrase suivante et la formation avec des mini-lots et des taux d'apprentissage beaucoup plus importants. Le

modèle RoBERTa a été proposé dans RoBERTa: A Robustly Optimized BERT Pretraining Approach par Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. Il est basé sur le modèle BERT de Google publié en 2018.

**ALBERT (A Lite BERT for Self-supervised Learning of Language Representations)** : il présente des techniques de réduction des paramètres pour réduire la consommation de mémoire et augmenter la vitesse d'entraînement de BERT. L'idée principale d'ALBERT est de réduire le nombre de paramètres (jusqu'à 90% de réduction) en utilisant de nouvelles techniques tout en ne prenant pas un gros coup à la performance. Maintenant, cette version compressée évolue beaucoup mieux que le BERT original, améliorant les performances tout en gardant le modèle petit.

### 3.7 Question-answering

La réponse aux questions (QA) est une discipline informatique dans les domaines de la recherche d'information et du traitement du langage naturel (NLP), qui s'intéresse à la construction de systèmes qui répondent automatiquement aux questions posées par les humains dans un langage naturel. Dans les systèmes de réponse aux questions, les embeddings jouent un rôle crucial en permettant au système de comprendre les relations sémantiques entre les mots dans les questions et les réponses. Cela permet au système de mieux comprendre les questions posées et de générer des réponses plus pertinentes. Dans la partie précédente, nous avons présenté en quelques mots les différents types d'embedding. Nous nous proposons dans cette partie de présenter plus en détails les modèles d'embedding Word2Vec et BERT.

- **Word2Vec**

Dans la figure suivante, les mots dans la case bleue sont appelés mots cibles et les mots dans les cases blanches sont appelés mots de contexte dans une fenêtre de taille 5.

Nous allons présenter le modèle CBOW ce qui signifie que le contexte est l'entrée de notre modèle et le mot cible (mot bleu) est la sortie.

L'intuition derrière ce modèle est assez simple : étant donné une phrase, nous choisirons notre mot cible pour être « Brown » et nos mots de contexte pour être [« The », « quick », « fox », « jumps »]. Ce modèle va prendre les représentations distribuées des mots du contexte pour essayer de prédire le mot cible.

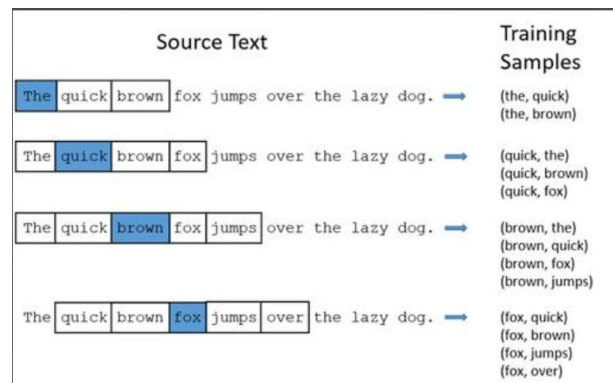


Figure 5 : Modèle CBOW

Intuitivement, le modèle Skip-Gram peut être considéré comme l'opposé du modèle CBOW. L'architecture de ce modèle prend le mot courant en entrée et essaie de prédire avec précision les mots avant et après ce mot. Ce modèle essaie essentiellement d'apprendre et de prédire des mots contextuels autour d'un mot d'entrée donné. En effet, les données d'apprentissage consistent en des combinaisons par paires de ce mot cible et de tous les autres mots dans la fenêtre. Il s'agit des données d'entraînement résultantes pour le réseau de neurones. Fondamentalement, une fois le modèle formé, il peut donner des probabilités qu'un mot soit le mot de contexte pour une cible donnée. Le schéma suivant montre l'architecture du réseau neuronal du modèle Skip-Gram.

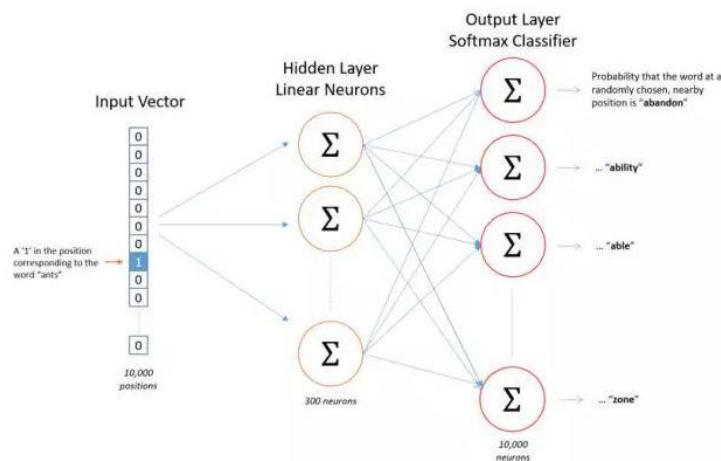


Figure 6 : Modèle Skip Gram

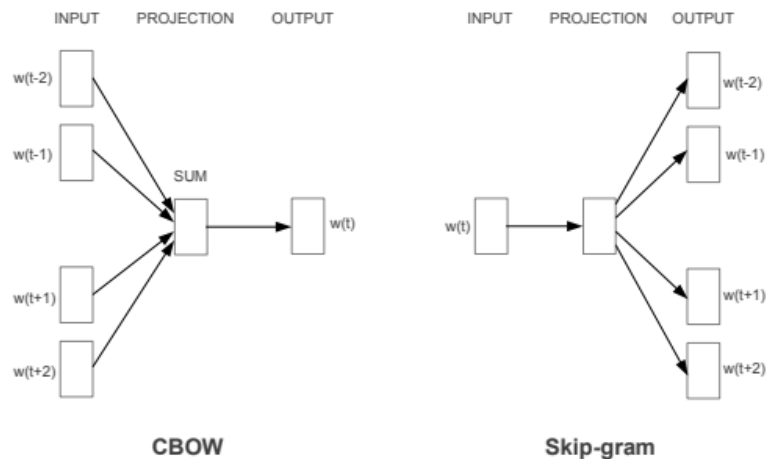


Figure 7 : Comparaison entre CBOW et Skip-gram

- **BERT**

BERT est un modèle de traitement du langage pré-entraîné qui utilise un procédé d'entraînement appelé "masquage de jeton" pour améliorer sa compréhension du contexte. Le modèle est entraîné sur un grand corpus de textes en utilisant cette méthode, qui consiste à masquer aléatoirement certaines parties du texte d'entrée, et à demander au modèle de prédire les jetons masqués en utilisant le contexte fourni par les jetons non masqués.

L'architecture de BERT se compose de plusieurs couches de blocs de transformer qui prennent en entrée des séquences de tokens (mots ou jetons prétraités) et les codent en représentations vectorielles. Ces couches sont suivies par des couches de normalisation de batch et de couches fully-connected pour produire des prédictions pour les tâches de traitement du langage.

BERT utilise un "masked language modeling" (MLM) pour l'entraînement, où un certain pourcentage des mots dans la séquence d'entrée sont masqués aléatoirement et le modèle doit prédire ces mots masqués en utilisant le contexte bidirectionnel de la séquence. Cela permet de BERT de comprendre le contexte bidirectionnel des mots dans une phrase, ce qui est crucial pour les tâches de traitement du langage.

Pour l'entraîner, on utilise un corpus de données de plusieurs milliards de phrases, en utilisant un grand nombre de paramètres pour optimiser les performances du modèle. L'entraînement peut prendre plusieurs jours, voire plusieurs semaines, selon la puissance de calcul disponible. Une fois l'entraînement terminé, le modèle est capable de comprendre le contexte dans lequel des mots ou des phrases sont utilisés, ce qui lui permet de répondre de manière plus précise à des questions ou de classer des textes de manière plus précise.



BERT étant un modèle pré-entraîné, il est possible de le réutiliser pour différentes tâches de traitement du langage en utilisant une technique appelée "fine-tuning" qui consiste à adapter le modèle pré-entraîné à un ensemble de données spécifique pour une tâche donnée.

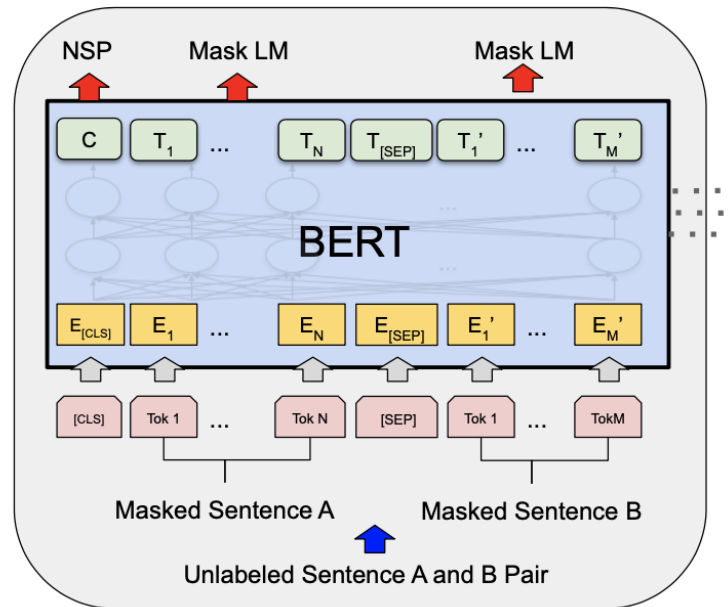


Figure 8 : Pré-entraînement BERT

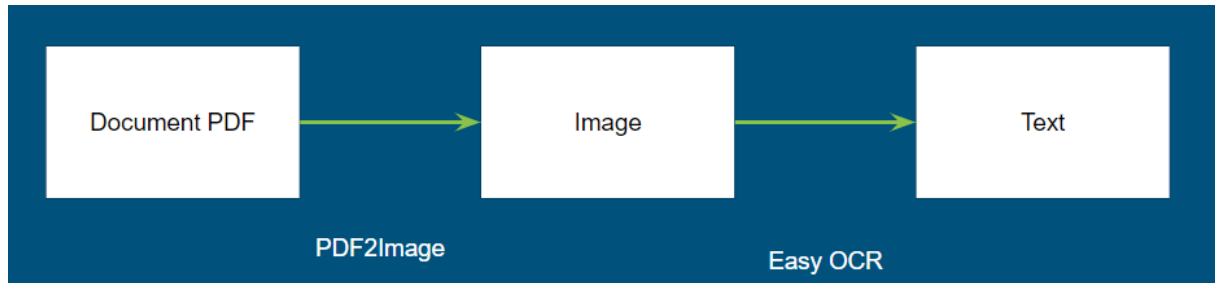
## 4. Processus de mise en place du chatbot

La mise en place d'un chatbot est un processus complexe qui nécessite une combinaison de différentes techniques de traitement automatique de la langue naturelle (NLP) pour permettre au chatbot de comprendre les questions des utilisateurs et de générer des réponses pertinentes. Les objectifs clés de ce processus comprennent l'extraction de texte à partir de différents formats de fichiers, la détermination des meilleures méthodes d'embedding, la transcription de fichiers audio ou vidéo et la recherche d'informations dans un grand nombre de documents. Pour ce faire, nous avons testé un ensemble de code que vous pouvez trouver via le lien suivant : <https://github.com/Achta00/Chatbot>

### 4.1 Extraction du texte dans des fichiers

Parmi les étapes les plus importantes dans la recherche des informations, l'extraction ou le scraping des données des fichiers étudiés de n'importe quel type, soit de format PDF, PPT ou DOCX est crucial.

Actuellement, le format le plus utilisé et populaire est le format PDF. En général, les documents déposés par les professeurs sur les plateformes LMS sont des PDF. Dans un premier temps, nous avons concentré notre étude sur ce format. Nous transformons le fichier PDF en image à l'aide de la bibliothèque PDF2Image puis nous utilisons un OCR nous permettant de transformer l'image en texte.



Ce pipeline est fonctionnel. Cependant, la consigne est de traiter plusieurs formats DOCX et PPT. Pour ne pas changer de méthodologie, nous utilisons les bibliothèques Python pour convertir des fichiers PowerPoint (pptx) et Word (docx) en fichiers PDF. Les bibliothèques habituellement utilisés sont PPTX2PDF et DOCX2PDF.

Les parties de changement d'extension étant triviales, il est à présent intéressant d'étudier davantage le mode de fonctionnement d'un OCR. Un OCR est une technologie qui reconnaît le texte dans une image numérique. Il est couramment utilisé pour reconnaître le texte dans les documents numérisés. Easy OCR est une bibliothèque Python pour la reconnaissance optique de caractères (OCR) qui permet de lire du texte à partir d'images et de PDF. Il est facile à utiliser. Voici les étapes à suivre pour utiliser Easy OCR pour extraire le texte d'une image :

- Installer Easy OCR en utilisant pip : `pip install easyocr`
- Importer la bibliothèque : `import easyocr`
- Charger le modèle de reconnaissance de caractères
- Afficher le texte extrait

## 4.2 Tests de quelques méthodes d'embedding

Comme expliqué précédemment, les embeddings sont des éléments cruciaux pour le dimensionnement d'un question-answering chatbot. La liste suivante présente les méthodes testées.

- Comparaison de la méthode de similarité avec Word2vec sur un fichier pdf
- Question-réponse avec BERT
- Question réponse avec CamemBERT

- Question-réponse avec BERT multilingues

### 4.2.1 Comparaison de la méthode de similarité avec Word2vec

Nous avons travaillé sur un code de question réponse avec les deux méthodes la similarité en cosinus et le modèle pré-entraîné de la méthode Word2Vec. La similarité en cosinus se basé sur la recherche de la similarité entre les embeddings de la question et chaque phrase de documents étudiés. En effet, la phrase qui présente la plus grande similarité avec la question est prédite comme la sortie du modèle pour la question donnée. Les images suivantes illustrent les résultats des deux méthodes :

```
[ ] print("Question: ", question)
    print("Answer: ", sentences[index])

Question: what are advantages of chatbot
Answer: chatbot short for chatterbot is an artificial intelligence ai feature that can be embedded and used through any major messaging application
```

---

```
▶ print("Question: ", question)
  print("Answer: ", cleaned_sentences_with_stopwords[index])


Question: what are advantages of chatbot
Answer: some examples of chatbot technology are virtual assistants like amazons alexa and google assistant and messaging apps such as wechat and facebook's messenger
```


### 4.2.2 Question réponse avec BERT

La question-réponse avec BERT utilise l'embedding de BERT pour comprendre les relations sémantiques entre les mots dans les questions et les réponses. Cela permet au système de mieux comprendre les questions posées et de générer des réponses plus pertinentes.

Pour résumer le code testé, nous utilisons un modèle de BERT et de tokenizer pré-entraîné à partir de la base de données SQUAD. Ils ne font aucune différence entre minuscule et majuscule.

```
▶ import torch
  from transformers import BertForQuestionAnswering
  model = BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
```

Downloading: 100%  443/443 [00:00<00:00, 9.77kB/s]

Downloading: 100%  1.34G/1.34G [00:40<00:00, 37.7MB/s]

---

```
[ ] from transformers import BertTokenizer
    tokenizer = BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
    # Téléchargement d'un tokenizer préentraîné
```

Pour un texte simple, nous créons un tenseur que nous limitons à 512 jetons. Ce qui nous permet de créer un dictionnaire avec les mots du texte. Nous ajoutons les jetons de début de séquence [CLS] et de séparateur [SEP]. Pour cela, nous utilisons la fonction torch.cat, qui concatène une

liste de tenseurs. A partir d'une lecture du texte par le modèle, nous déterminons la réponse la plus probable. Après analyse avec les méthodes précédentes, le modèle BERT nous permet d'avoir de meilleur résultat. En effet, dans les seuls, il est le seul à avoir trouvé une réponse correcte car celle-ci avait plus de cohérence avec la question. Pour les autres, on obtenait juste une phrase du texte avec l'élément clé de la question.

Pour la suite des textes, une fois le modèle BERT testé, tester ses dérivés est une démarche simple à effectuer. En effet, il suffit juste de remplacer le modèle et le tokenizer sans pour autant changer les autres lignes de code.

#### 4.2.3 *Question réponse avec CamemBERT*

CamemBERT est un modèle de traitement du langage développé basé sur le modèle BERT, mais qui est pré-entraîné sur un corpus de données en français. Il est conçu pour comprendre le contexte bidirectionnel des mots dans une phrase, ce qui est crucial pour les tâches de traitement du langage telles que la compréhension de la langue naturelle et la réponse aux questions. Pour tester un modèle de réponse aux questions basé sur CamemBERT, nous avons utilisé comme base de données le SQuAD français : FQuAD . Dans l'ensemble, les tests de questions réponses avec des modèles dérivés de BERT son tous les mêmes. Il faut juste changer la base de donné et télécharger le bon modèle en fonction de l'application. Ainsi, pour tester ce modèle, nous avons utilisé le modèle BERT en remplaçant la base de données et le type de modèle par les lignes de codes suivantes :

```
import torch
from transformers.modeling_camembert import CamembertForQuestionAnswering
model_path = 'fmikaelian/camembert-base-fquad'
model = CamembertForQuestionAnswering.from_pretrained(model_path)
```

#### 4.2.4 *Question réponse avec BERT multilingue*

C'est un modèle de transformers pré-entraîné de manière auto-supervisée sur un large corpus de données multilingues. Cela signifie qu'il a été exécuté uniquement sur du texte brut, sans que les humains les étiquettent de quelque manière que ce soit, et en utilisant un processus automatisé pour générer des entrées et des étiquettes à partir de ces textes.

Ce modèle est préformé à deux objectifs :

- Modélisation du langage caché (MLM) : en prenant une phrase, le modèle cache au hasard 15 % des mots de l'entrée, puis exécute la phrase cachée entière à travers le modèle et doit prédire les mots cachés. Il permet au modèle d'apprendre une représentation bidimensionnelle de la phrase.

- Prédiction de la suivante phrase (NSP) : modèles qui joignent deux phrases masquées en entrée lors de la pré-préparation. Parfois, ils correspondent à des phrases qui sont côte à côte dans le texte original, parfois non. Ensuite, le modèle doit prédire si les deux phrases sont consécutives oui ou non.

Comme expliqué précédemment, nous avons utilisé le programme de BERT mais en changeant le modèle et le Tokenizer en un Bert-multilingues.

```
import torch
from transformers import BertForQuestionAnswering
model = BertForQuestionAnswering.from_pretrained('bert-base-multilingual-cased')
```

Figure 9: le code de Bert-multilingual

```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
# Téléchargement d'un tokenizer préentraîné
```

Figure 10: Bert\_multilingual pour le Tokenizer

Nous avons constaté l'absence des lettres sur la réponse due à l'extraction des PDF en texte.

```
answer_question(question1, pdf_txt)
# On applique la fonction sur la question et le texte pour obtenir une réponse

Query has 493 tokens.

(' What is chatbot?',
 '##bot ##s are also limited in the scope of quer ##ies that they are able to respond to',
 0.9859137535095215)
```

Figure 11: Résultat de test

En somme, ces codes permettent de tester différentes méthodes d'embedding pour les systèmes de question-réponse et de déterminer laquelle est la plus efficace pour une tâche donnée.

### 4.3 Question-réponse avec cdQA sur un ensemble de fichiers

Closed-Domain Question Answering (CDQA) est un type de système de réponse aux questions qui est limité à une certaine "domaine fermé" de connaissances. Cela signifie que le système de réponse aux questions ne peut répondre qu'à des questions qui peuvent être répondues à l'aide des informations contenues dans un corpus de données spécifique, comme un ensemble de données de documents, un site web ou une base de données.

Le fonctionnement des systèmes de CDQA est généralement basé sur l'indexation de ces données, suivi d'une recherche pour trouver les informations pertinentes pour répondre à la question. Ils utilisent des techniques de traitement automatique du langage naturel pour comprendre les questions et identifier les informations pertinentes dans les documents. Toutefois, ces systèmes sont limités par la qualité et la quantité des données disponibles, ils peuvent donc ne pas être capables de répondre à toutes les questions qui peuvent être posées sur un sujet donné.

Pour tester un système CDQA, nous avons utilisé un ensemble de données de questions-réponses spécifiques à un domaine pour évaluer les performances du système. Nous nous sommes basées sur des fichiers basés sur des comptes rendus de l'entreprise Amazon. Pour obtenir de meilleurs résultats, nous avons observé qu'il était important de choisir un ensemble de données qui contient des questions qui peuvent être répondues à l'aide des informations contenues dans le corpus de données utilisé par le système.

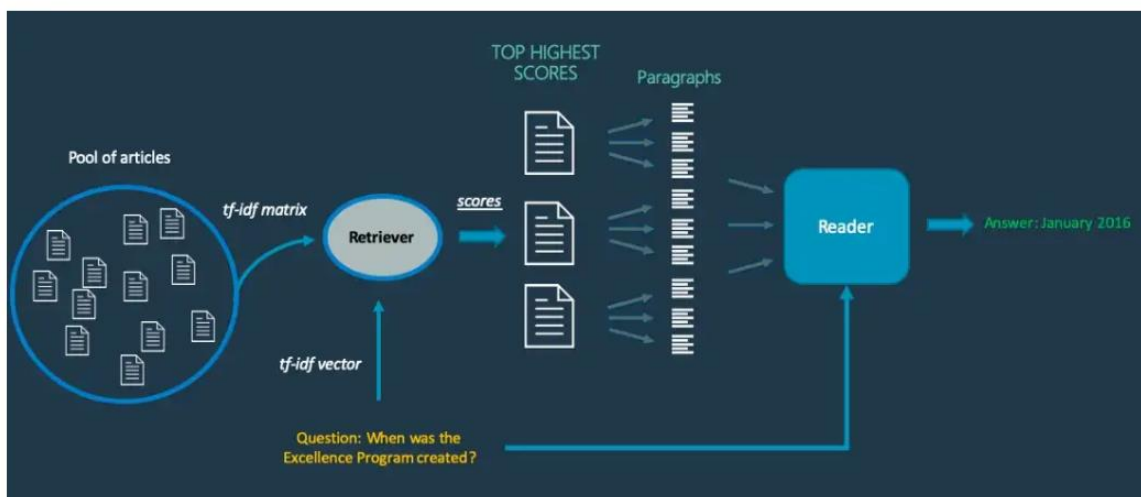


Figure 12 : Architecture cdQA

Nous avons testé un code dont la base de données des fichiers est un fichier CSV qui contient la date, le titre, le lien d'article bnp accessible sur internet. Le package CDQA contient des packages permettant de convertir des fichiers PDF en texte interrogeable dans des blocs de données. Nous avons appelé le pipeline pour définir le reader et retriever pour la recherche d'informations.

```
cdqa_pipeline = QAPipeline(reader='./models/bert_qa.joblib')
cdqa_pipeline.fit_retriever(df=df)
```

Ensuite nous avons commencé à poser des questions et les réponses seront prédites.

```
query = 'Since when does the Excellence Program of BNP Paribas exist?'  
prediction = cdqa_pipeline.predict(query)
```

## 4.4 Haystack

### 4.4.1 *Présentation et fonctionnement*

Haystack est un projet open-source développé par la société DeepSet qui vise à faciliter la recherche de documents et de données structurées. Il utilise des techniques d'apprentissage automatique pour comprendre le contenu des documents et les mettre en correspondance avec les requêtes de recherche des utilisateurs. Haystack prend en charge plusieurs formats de données, tels que des fichiers PDF, Word, Excel, etc. et peut être utilisé pour une variété de tâches de recherche, telles que la recherche de documents juridiques, la recherche de données d'entreprise, etc.

Il est aussi conçu pour la tâche de réponse à des questions (Question Answering, QA). Il utilise des techniques d'apprentissage automatique pour comprendre les questions posées par les utilisateurs et trouver les réponses les plus pertinentes dans les documents indexés. De plus, il utilise des modèles de compréhension de langue naturelle pour comprendre les questions et des modèles de recherche pour trouver les réponses les plus pertinentes dans les documents. Son fonctionnement repose sur 3 éléments : les nœuds, les pipelines et REST API.

- Les nœuds sont des blocs de construction qui traitent et acheminent l'information. Ils existent plusieurs types de nœuds enchainés les uns avec les autres à l'aide d'un pipeline.
- Un pipeline peut être symbolisé par un tuyau pour le transport d'un élément. Sur Haystack , un pipeline est la succession de nœuds pour la création de scénario de recherche de réponses aux questions.
- REST API est un service qui peut rester actif pour gérer les demandes. Une fois configuré, il peut interagir avec des pipelines via des requêtes HTTP et connecter Haystack à une interface graphique orientée utilisateur.







#### 4.4.2 Test

Après avoir compris le fonctionnement de Haystack, nous avons testé un code du Haystack dans deux versions en français et en anglais. Il s'agit d'apprendre à configurer un système de questions-réponses capable de rechercher dans des bases de connaissances complexes dans Elasticsearch et de mettre en évidence les réponses à des questions telles que "c'est quoi l'intelligence artificielle ?".

Pour effectuer nos tests, nous avons créé une base de données avec 15 fichiers en format txt. En effet, Haystack traite unique des fichiers txt . Toutefois, il dispose de bloc permettant la conversion d'autres types de fichiers tels que PDF, DOCX, PPTX, ODT , etc en TXT.

Au début nous avons les stocker dans un fichier zip, ensuite nous avons dézippé les fichiers dans la base de données afin d'être utile dans le Elasticsearch.

Puis on installe la dernière version de Haystack :

```
# Install the latest release of Haystack in your own environment
#! pip install farm-haystack

!pip install --upgrade pip
!pip install git+https://github.com/deepset-ai/haystack.git#egg=farm-haystack[colab]
```

Figure 14: Installation de Haystack

Notre Retriever a pour but de trier les documents et de ne sélectionner que ceux qui sont pertinents pour la question posée. Pour cela, nous avons utilisé l'algorithme BM25 dans notre code.

```
from haystack.nodes import BM25Retriever

retriever = BM25Retriever(document_store=document_store)
```

Figure 15: Initialisation du Retriever

Après cette étape, nous avons mis en place un Reader qui analyse les textes sélectionnés par le Retriever et en extrait les réponses les plus appropriées. Pour la version anglaise, nous utilisons le FARMReader avec le modèle RoBERTa de base deepset/roberta-base-squad2. Pour la

version française, nous avons modifié le Reader pour utiliser CamemBert afin de tester les performances de notre système.

```
reader = FARMReader(model_name_or_path="deepset/roberta-base-squad2", use_gpu=True)
```

Figure 16: Haystack version anglaise

```
reader = FARMReader(model_name_or_path="etalab-ia/camembert-base-squadFR-fquad-piaf", use_gpu=True)
```

Figure 17: Haystack version française

Ensuite nous avons créé un pipeline Retriever-Reader ExtractiveQAPipeline pour lier les deux.

```
from haystack.pipelines import ExtractiveQAPipeline  
  
pipe = ExtractiveQAPipeline(reader, retriever)
```

Figure 18: Création du Pipeline

Après l'exécution des commandes précédentes, nous posons une question de telle sorte que le retriever cherche parmi les 10 meilleurs documents et le Reader affiche les 5 meilleurs réponses.

```
# You can configure how many candidates the Reader and Retriever shall return  
# The higher top_k_retriever, the better (but also the slower) your answers.  
prediction = pipe.run(  
    query=query1, params={"Retriever": {"top_k": 10}, "Reader": {"top_k": 5}}  
)
```

Figure 19: Commande de la prédiction

```
from pprint import pprint  
  
pprint(prediction['answers'])
```

```
[<Answer {'answer': 'Problem solving', 'type': 'extractive', 'score': 0.7397282123565674, 'context':  
<Answer {'answer': 'the ability of a digital computer or computer-controlled robot to perform tasks  
<Answer {'answer': 'building machines that are intelligent', 'type': 'extractive', 'score': 0.50798:  
<Answer {'answer': 'bottom-up AI', 'type': 'extractive', 'score': 0.42558157444000244, 'context': 'a  
<Answer {'answer': 'replicate or simulate human intelligence in machines', 'type': 'extractive', 'sc
```

Figure 20: Résultat du code

#### 4.4.3 L'influence des stops words dans la prédiction des réponses

Les mots de liaison, également appelés "stop words", sont des mots couramment utilisés dans une langue qui ne contiennent généralement pas d'informations sémantiques significatives. Ils peuvent inclure des articles, des conjonctions et des prépositions, par exemple. Dans les modèles de question-réponse tels que Haystack, l'utilisation de ces mots de liaison peut avoir un impact important sur la précision des résultats. Toutefois, pour avoir des certitudes sur l'influence des stops words.

Nous avons trouvé trois méthodes pour supprimer les mots vides :

- Le package stopwords de la bibliothèque NLTK est une liste pré-construite de mots de liaison pour plusieurs langues, y compris le français, l'anglais, l'espagnol et plus encore. Ce package peut être utilisé pour filtrer les mots de liaison d'un texte donné en utilisant la liste de mots de liaison pour une langue particulière. Le package est facile à utiliser et peut être importé en utilisant le code suivant: *from nltk.corpus import stopwords*
- Le package get\_stop\_words de la bibliothèque stop\_words est un autre package qui contient une liste de mots de liaison pour plusieurs langues. Ce package peut être utilisé pour filtrer les mots de liaison d'un texte donné en utilisant la liste de mots de liaison pour une langue particulière. Le package est également facile à utiliser et peut être importé en utilisant le code suivant: *from stop\_words import get\_stop\_words*
- La méthode remove\_stpwrds() de la bibliothèque textcleaner est une méthode qui peut être utilisée pour supprimer les mots de liaison d'un texte donné. Cette méthode peut être importée et utilisée de la manière suivante: *from textcleaner import remove\_stpwrds*

```
# Chargement des mots vides en français
nltk.download("stopwords")
stop_words = set(stopwords.words("french"))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
# Fonction pour supprimer les mots vides
def remove_stop_words(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return " ".join(filtered_words)
```

```
# Texte à traiter
question="C'est quoi l'intelligence artificielle?"
```

Après avoir supprimé les stops words, nous affectons la question sans mots vides pour la prédiction.

```
# You can configure how many candidates the Reader and Retriever shall return
# The higher top_k_retriever, the better (but also the slower) your answers.
prediction = pipe.run(
    query=filtered_question, params={"Retriever": {"top_k": 10}, "Reader": {"top_k": 5}}
)
```

Query: une définition de l intelligence artificielle?

```
{  'content': 'ChatGPT, Midjourney... les IA sont plus que jamais dans '
      'l'actualité. Mais, quelle est la définition d'...',
  'meta': {'name': 'fichier_5.txt'},
  'name': 'fichier_5.txt'}

{  'content': "L'intelligence artificielle (IA) est un « ensemble de théories "
      'et de techniques mises en œuvre en vu...',
  'meta': {'name': 'fichier_1.txt'},
  'name': 'fichier_1.txt'}
```

Figure 21 : QA avec mots vides

Query: definition intelligence artificielle?

```
{  'content': 'ChatGPT, Midjourney... les IA sont plus que jamais dans '
    'l'actualité. Mais, quelle est la définition d'...',
  'meta': {'name': 'fichier_5.txt'},
  'name': 'fichier_5.txt'}

{  'content': "L'intelligence artificielle exploite les ordinateurs et les "
    'machines pour imiter les fonctions de ré...',
  'meta': {'name': 'fichier_2.txt'},
  'name': 'fichier_2.txt'}
```

Figure 22 : QA sans mots vides

Avec ce test, nous constatons que l'ordre des réponses n'est plus le même. La réponse avec le meilleur score dans les cas est la même mais la deuxième meilleure réponse change mais les réponses proposées sont pertinentes car en lien avec notre question.

Ayant effectué plusieurs test, nous avons constaté que l'inclusion de mots vides dans une requête de recherche peut réduire la qualité des résultats de recherche en augmentant le nombre de documents qui ne correspondent pas directement à la question. Durant ces tests, notre modèle étant un modèle simple, lorsque la question est longue, il est difficile d'obtenir une réponse précise. Dans certains cas, supprimer les mots vides nous permettait d'avoir de meilleurs résultats. Dans les tests effectués en anglais la suppression se fait aisément. Cependant, nous avons remarqué que dans les tests en français, quelques fois la question avec et sans mots vides étaient la même exemple '*Qu'est ce que c'est l'intelligence artificielle ?*'.

Pour nous '*Qu'est ce que*' est mot vide mais le modèle ne l'a pas considéré ainsi. Pour régler cela une liste de mots peut-être créée pour y inclure des mots vides.

Après avoir testé plusieurs codes, nous avons validé la solution Haystack comme étant celle la plus adaptée à notre application. En effet, au début du projet, nous avons implémenté différentes méthodes à partir des différentes librairies python. Or Haystack nous permet de réaliser les différentes opérations voulues. Elle présente plusieurs avantages :

- C'est une solution 'clé en main', étant un open source pour la mise en œuvre des systèmes de réponses aux questions, elle intègre des nœuds pour les différentes tâches que nous souhaitons réaliser tels que le moteur de recherche, les nœuds pour la conversion des fichiers et des pipelines faits pour la réponse aux questions
- Evite un gros développement en interne, les tutoriels sont facilement accessible sur GitHub et il présente une forte communauté
- Un vaste choix de modèle pour le pipeline de recherche
- Adaptation possible à plusieurs langues

## Evaluation

Dans le cadre du PRT, nous avons réalisé plusieurs recherches pour répondre au cahier des charges. En effet, pour le dimensionnement de ce chatbot, nous avons trouvé plusieurs méthodes et certaines n'étaient pas adaptées à notre application. Aussi, au début du PRT, nous avons comme été jetées dans un océan d'informations. Et ayant acquis plus d'expérience dans le domaine du langage naturel, nous sommes en mesure d'émettre un avis critique sur nos recherches pour approfondir et améliorer le modèle.

Dans la partie extraction du texte dans un fichier, nous avons constaté que la chaîne (PDF-IMAGE- TEXTE) fonctionne correctement. Et pour adapter ce modèle aux autres types de fichiers, nous avons effectué des conversions. Cela n'est pas l'idéal car nous avons une certaine pertes d'informations. Aussi, nous avons constaté que cette méthode permet d'extraire aisément le texte. Mais, l'extraction des figures, tableaux et formules à partir d'un OCR présente des défis. L'OCR est généralement plus efficace pour l'extraction de texte, car il est conçu pour reconnaître les caractères dans des images de documents. Cependant, l'extraction de figures, de tableaux et de formules peut être plus difficile, car ces éléments peuvent être présentés de manière différente dans différents documents et peuvent être plus difficiles à reconnaître pour l'OCR. Toutefois, les étudiants du CESI ayant travaillé sur une plus grande base de données ont obtenue la même conclusion que nous en utilisant les méthodes fournies par Haystack. Nous avons conclu, que l'étape de pré-processing des données étaient hyper importante de telle sorte a augmenté les performances du modèle.

Nous avons présenté quelques méthodes d'embedding à utiliser pour les systèmes de réponses aux questions. Toutefois, il manque à notre étude une certaine comparaison entre les modèles. En effet, nous avons trouvé que le calcul de score est un travail fastidieux car pour obtenir les meilleurs résultats, il est primordial de tester les modèles sur une large base de données dans des domaines différents.

Pour ces tests sur une grande base de données, il faut étiqueter les réponses. Il existe plusieurs méthodes pour le faire :

- Utiliser un corpus de données pré-étiqueté disponibles en ligne pour des tâches de question-réponse, comme SQuAD, NewsQA, etc. Ces corpus contiennent des questions et des réponses correspondantes étiquetées manuellement pour vous permettre de former et évaluer des modèles de question-réponse.
- Créer son ensemble de données en collectant des questions et en trouvant les réponses correspondantes manuellement. Cela peut être fastidieux et coûteux en termes de temps, mais cela peut garantir que les données sont pertinentes pour votre domaine d'application spécifique.

Dans notre cas, nous avons effectué des tests sur différents corpus, comme expliqué précédemment, lorsque le texte comportait des figures, tableaux ou formules les réponses du modèle ne correspondaient pas aux résultats attendues. Mais lorsque nous avons utilisé des fichiers qui comprenaient principalement du texte, les réponses étaient plus pertinentes. Nous avons calculer les scores manuellement pour passer cette étape d'étiquetage des données. Nous obtenons un score f1 d'approximativement 80% pour les modèles BERT et CamemBERT lorsque le fichier est composé principalement de texte. Pour le modèle BERT multilingue le score était d'environ 40%, lorsqu'on switch entre question en français et en anglais, on constate que en français, il y'a une perte des données car, les accents et autres caractères spéciaux sont supprimés.

## Conclusion

En somme, au travers des différents tests effectués de l'extraction de données au tests d'embedding en passant par le cdQA avec Haystack, nous avons permis au étudiants du CESI d'avoir une base pour l'implémentation du chatbot sur la plateforme Rainbow Classroom.

La solution finale présenté par les étudiants en informatique a été jugé satisfaisante par l'entreprise. A partir de ces travaux, les ingénieurs d'Alcatel pourront peaufiner le travail en améliorant notamment l'extraction des données dans un fichier de telle sorte à ce que le bot puisse traiter du texte, des images, des formules ...

Au terme de ce projet d'étude pour la création d'un chatbot pour plateforme LMS et Rainbow Classroom, nous avons intégré un grand nombres de notions dans le traitement automatique du langage naturel. Aussi, nous avons été motivées pour effectuer ce travail car en tant qu'étudiantes nous pensons qu'à terme ce projet permettra aux étudiants et professeurs de travailler plus efficacement.