



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université des Sciences et de la Technologie Houari Boumediene**

**Faculté d'Electronique et d'Informatique  
Département Informatique**

Mémoire de Master

Filière: Informatique

Spécialité:

Systèmes informatiques  
intelligents (SII)

---

## **Thème**

**Conception d'un système de questions réponses  
basé documents par apprentissage automatique**

---

**Sujet Proposé et encadré par :**

BENSAOU Nacéra

DJAOUEDA Moussa Nadjib

**Présenté par :**

Dahimene Ouassil

Messaouali Abdeslem

**Soutenu le : .. / .. /....**

**Devant le jury composé de:**

**M..... Président (e)**

**M..... Membre**

# Remerciements

*La réalisation de ce projet de fin d'étude a été possible grâce à la participation directement ou indirectement de personnes en qui nous voulons leurs témoigner toute notre gratitude. Nous tenons tout d'abord à remercier nos parents et nos familles ainsi que nos proches amis pour tous les sacrifices, l'amour et le soutien apporté de leurs part.*

*Nos plus sincères remercîments et reconnaissances vont ensuite directement envers notre promotrice madame Bensaou et à notre co-promoteur monsieur Djaouda pour leurs conseils judicieux, leur patience, leur disponibilité ainsi que les connaissances et outils nécessaires qui ont longuement contribués à alimenter notre réflexion.*

*A Toute personne qui a contribué de près ou de loin pour qu'on en arrive à ce stade de notre chemin universitaire, nous leurs disons merci et que dieu leurs donne en retour les fruits de leurs bonnes volontés.*

*Dahimene Ouassil & Messaouali Abdeslem*

# TABLE DES MATIÈRES

<b>Introduction</b>	<b>7</b>
<b>1 Préliminaires</b>	<b>9</b>
1.1 Les Systèmes de réponses aux questions (Question Answering) . . . . .	9
1.2 Historique . . . . .	10
1.2.1 Définition et types d'une question : . . . . .	11
1.3 Domaine ouvert et Domaine fermé . . . . .	13
1.3.1 Domaine fermé . . . . .	13
1.3.2 Domaine Ouvert . . . . .	13
1.4 Types de QA systèmes . . . . .	14
1.4.1 Systèmes basé Texte (Text-Based) : . . . . .	14
i        Basé Web (Web-Based) : . . . . .	14
ii      Basé Document (Document-based) : . . . . .	14
iii     Basé recherche d'informations (IR-Based) : . . . . .	15
1.4.2 Basé Connaissances (Knowledge-based) . . . . .	16
1.4.3 Les systèmes hybrides . . . . .	17
1.4.4 Les modèles Deep Learning: . . . . .	17
1.5 Les systèmes fréquents d'application du question answering: . . . . .	17
1.6 L'apprentissage profond . . . . .	18
1.6.1 Word embeddings . . . . .	19
1.6.2 . . . . .	21
i        Réseaux de neurones récurrents (RNN) . . . . .	21
ii      Le problème de la disparition du gradient dans les RNN . . . . .	23
iii     Long short-term memory (LSTM) et Gated Recurrent Unit (GRU) . . . . .	24
1.6.3 Modèle de séquence à séquence (Seq2Seq model) . . . . .	26
1.6.4 Mécanisme d'attention . . . . .	28
1.6.5 Architecture Transofrmer : . . . . .	29
1.6.6 L'apprentissage par transfert . . . . .	31

<b>2 Conception</b>	<b>34</b>
2.1 Stanford Question Answering Dataset (SQuAD) . . . . .	34
2.1.1 SQuAD : Un dataset de questions factuelles . . . . .	34
i        Quelques détails sur le dataset . . . . .	34
ii      Quelques statistiques descriptives sur le dataset . . . . .	35
2.1.2 Prétraitements sur le dataset . . . . .	37
i        Nettoyage du dataset . . . . .	37
2.2 Modèles proposés . . . . .	40
2.2.1 Modèle de distance . . . . .	40
2.2.2 Modèle basé LSTM (avec et sans mécanisme d'attention) : . . . . .	45
i        Partie 1 : Word embeddings . . . . .	45
ii      Partie 2 : Extraction des vecteurs de contextes . . . . .	45
iii     Partie 3 : Matrice de similarité et mécanisme d'attention . . . . .	45
iv     Partie 4 : Concaténation globale et Couches de sortie . . . . .	48
v      L'entraînement . . . . .	48
2.2.3 Modèle d'apprentissage par transfert (transfer learning) . . . . .	51
2.3 Résultats d'entraînement . . . . .	53
i        Modele LSTM sans attention . . . . .	53
ii      Modele LSTM avec attention : . . . . .	54
iii     Modele par apprentissage par transfert . . . . .	55
<b>3 Implémentation et tests</b>	<b>57</b>
3.1 Environnement de développement et d'expérimentation . . . . .	57
3.1.1 Plateformes, outils et langages de programmation . . . . .	57
3.2 Tests et résultats . . . . .	59
3.2.1 Corpus de tests . . . . .	59
3.2.2 Mesures de performance . . . . .	59
3.2.3 Résultats des tests . . . . .	59
3.3 Interface de l'application . . . . .	64
<b>Conclusion</b>	<b>68</b>

# TABLE DES FIGURES

1.1	Architecture de base d'un système QA . . . . .	10
1.2	Architecture de base d'un système QA basé Web . . . . .	14
1.3	Architecture de base d'un système QA basé document . . . . .	15
1.4	Architecture de base d'un système QA basé Recherche d'information . . . . .	16
1.5	Architecture de base d'un système QA basé connaissances . . . . .	17
1.6	Exemple d'utilisation de google assistant . . . . .	18
1.7	Exemple de réponse QA de google . . . . .	18
1.8	Architectures de CBOW et Skipgram . . . . .	19
1.9	Exemple des relations entre les mots en utilisant Glove. . . . .	20
1.10	Exemples de représentations avec Glove . . . . .	21
1.11	Schéma basique d'une architecture de réseaux de neurones récurrents. . . . .	22
1.12	Schéma d'une cellule d'un réseaux de neurones récurrents . . . . .	22
1.13	Types d'utilisation de réseaux de neurones récurrents . . . . .	23
1.14	Schéma d'une cellule LSTM . . . . .	25
1.15	Schéma d'une cellule GRU . . . . .	25
1.16	Schéma simplifié d'un modèle de séquence à séquence (seq2seq) . . . . .	26
1.17	Schéma d'un modèle de séquence à séquence (seq2seq) . . . . .	27
1.18	Exemple d'un modèle seq2seq en traduction automatique . . . . .	27
1.19	Exemple d'attention en traduction automatique . . . . .	28
1.20	Architecture de base d'un modèle Transformer . . . . .	30
1.21	Schéma représentant une multicouche d'attention d'un Transformer . . . . .	30
1.22	Comparaison de l'apprentissage traditionnel avec l'apprentissage par transfert . . . . .	32
1.23	Principe de base du réglage fin (Fine-tuning) . . . . .	32
2.1	Fréquences des textes par rapport à leur taille . . . . .	36
2.2	Distributions des <i>Wh-questions</i> dans le dataset SQuAD. . . . .	37
2.3	Exemple d'un texte qui contient des mots étrangers . . . . .	38
2.4	Fonctionnement du modèle InferSent de FACEBOOK . . . . .	42
2.5	Architecture générale du modèle de Question Answering basé sur la distance . . . . .	44
2.6	Schéma représentant la construction de la matrice de similarité. . . . .	46
2.7	Exemple de matrice de similarité . . . . .	47

2.8 Schéma de constructions des deux matrices d'attention . . . . .	48
2.9 Architecture du modèle LSTM sans attention . . . . .	49
2.10 Architecture du modèle LSTM avec attention . . . . .	50
2.11 Architecture de BERT (Multi-couches d'encodeurs de Transformer) . . . . .	51
2.12 Entrées et sorties de BERT . . . . .	51
2.13 Schéma de la partie embeddings de BERT . . . . .	52
2.14 Nouveau modèle adapté au Question Answering en utilisant BERT . . . . .	53
2.15 Courbes d'erreur pendant l'entraînement et le test . . . . .	54
2.16 Courbes d'erreur pendant l'entraînement et le test . . . . .	55
2.17 Courbes d'erreur pendant l'entraînement et le Test . . . . .	56
3.1 Logos de quelques outils utilisés . . . . .	58
3.2 Pourcentage de bonnes réponses par type de questions . . . . .	61
3.3 Matrice d'attention du modèle LSTM avec attention . . . . .	62
3.4 Matrice d'auto attention (self attention) du modèle BERT . . . . .	62
3.5 Temps moyen d'exécution en secondes par modèle . . . . .	63
3.6 Interface de la page d'accueil . . . . .	64
3.7 Sélectionner un texte pour le test . . . . .	65
3.8 Test de réponse sur l'interface . . . . .	65
3.9 Affichage de quelques détails d'une réponse . . . . .	66
3.10 Affichage d'une matrice d'attention sur l'interface . . . . .	66
3.11 Affichage du tableau des distances du modèle de distances . . . . .	67

# INTRODUCTION

Nous vivons dans une période où la technologie a pris le dessus sur notre quotidien, notre utilisation devient de plus en plus complexe et exigeante, la quantité de données disponible partout est phénoménale et l'homme s'y perd pour chercher des réponses. Ainsi, pour arriver à satisfaire les besoins de l'homme, il faut d'abord arriver à comprendre l'homme, comprendre ce qu'il cherche, comprendre comment il cherche ses réponses. Il faut ainsi donc comprendre le langage naturel, et le meilleur moyen de s'assurer avoir compris un texte est de répondre à des questions en rapport avec celui-ci.

Parmi les problèmes les plus étudiés récemment en traitement automatique du langage naturel (TALN) celui de concevoir un système automatique de questions-réponses qui consiste à répondre à des questions posées en langage naturel sur des sources données. C'est l'un des plus anciens domaines de recherche, il est appliqué dans une grande variété de tâches, telle que la recherche d'information, l'extraction d'entités mais aussi des systèmes de dialogues conçus pour simuler une conversation humaine tels que les assistants virtuels comme celui de Google, Siri d'Apple ou bien Alexa d'Amazon.

De nombreux chercheurs se sont penchés sur ce problème. On peut ainsi trouver plusieurs approches des plus traditionnelles comme les systèmes experts à bases de règles et de faits, ou les techniques conventionnelles du TALN comme l'extraction d'entités nommées et l'analyse syntaxique, mais encore des méthodes de recherche d'information à base de distances et similarités. Néanmoins, avec les récents développements de l'apprentissage automatique et l'apprentissage profond, les modèles qui en résultent se sont révélés très prometteurs pour les systèmes de questions-réponses, bien qu'ils nécessitent généralement beaucoup plus de données et d'entraînement que les approches traditionnelles.

Il existe plusieurs corpus spécialement conçus pour la compréhension de texte à base de questions-réponses, tels que SQuAD (Stanford Question Answering Dataset) ou bien WikiQA (Wikipedia Question Answering), la plupart des corpus disponibles sont pour la langue anglaise mais il en existe quelques uns dans d'autres langues.

Ce projet consiste à proposer une ou plusieurs approches pour concevoir et implémenter un

système de questions-réponses basé document par apprentissage automatique, c'est-à-dire créer un modèle capable de comprendre un texte provenant d'un document donné et de répondre à des questions en rapport avec celui-ci. Nous considérons des documents et corpus traités en anglais sur des questions factuelles.

En dehors de cette introduction, ce mémoire est organisé en trois chapitres :

- Le premier chapitre est divisé en deux parties, dans la première nous présentons le Question Answering, sa définition, son historique, et les architectures des différentes solutions proposées à ce problème par rapport aux techniques de résolution de problème ainsi qu'aux types de questions traitées. La deuxième partie de ce chapitre contient les définitions, les notions et les outils théoriques nécessaires à la compréhension du travail.
- Dans le deuxième chapitre, nous expliquons la conception des trois solutions que nous proposons, en détaillant les composantes principales de chacune.
- Le dernier chapitre est réservé à l'aspect pratique de notre travail. Nous présentons les différentes technologies que nous avons utilisée lors de la conception et le développement de nos solutions et de notre application. Le reste de ce chapitre est consacré aux résultats des tests et à leur discussion.

Nous achérons la présentation de ce travail par une conclusion dans laquelle nous proposons quelques pistes de travail de notre problématique qui pourront améliorer le projet.

## CHAPITRE

# 1

# PRÉLIMINAIRES

Dans ce chapitre nous présentons tous les concepts, les notions et éléments nécessaires à la compréhension de ce travail notamment sur le *Question Answering* et quelques modèles d'apprentissage automatique que nous déployons dans notre solution.

## 1.1 Les Systèmes de réponses aux questions (*Question Answering*)

*Question answering* (QA) est une discipline du traitement automatique du langage (TALN) qui consiste à créer un système qui répond automatiquement à des questions posées en langage naturel.

Les systèmes QA peuvent être implémentés en différentes architectures et approches. Les données utilisées pour construire la réponse peuvent être structurées comme les bases de données ou bien non structurées comme des documents écrits dans un langage naturel. Ils peuvent être des documents texte, des images, des pages web etc...

Le challenge est de pouvoir répondre à plusieurs types de questions des plus simples aux plus complexes, aux questions posées dans un domaine spécifique (Closed-Domain) ou bien dans n'importe quel domaine (Open-Domain). La réponse doit être la plus précise possible selon le type de questions.

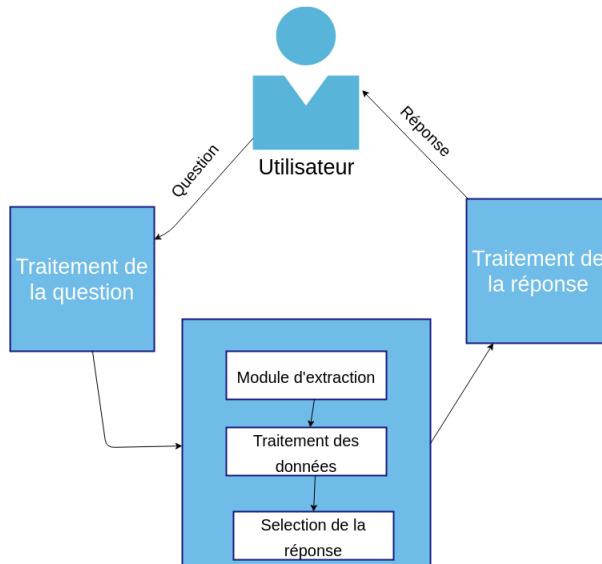


FIGURE 1.1 – Architecture de base d'un système QA

## 1.2 Historique

La recherche en rapport avec les systèmes de QA a commencé dès le début des travaux en Intelligence Artificielle dans les années 60. En effet, à cette époque il existait des systèmes experts bien spécifiques à des domaines qui pouvaient gérer des requêtes bien spécifiques ; mais cela restait très limité par rapport aux travaux du traitement de langage naturel dans le domaine qui se développait.

Parmi les tous premiers systèmes de questions-réponses il y a BASEBALL[1] développé en 1961 qui pouvait répondre à des questions sur la ligue américaine de baseball durant une période d'une année. Il y a aussi LUNAR[2] développé en 1971 qui est basé sur les analyses géologiques des roches apportés par la mission *Apollo*.

Les deux systèmes étaient précis dans leurs domaines, par exemple LUNAR était capable de répondre à 90% des questions en langage naturel posées par des professionnels du domaine. Plusieurs autres systèmes ont très vite suivi, la plupart avaient une architecture qui s'appuyait sur des bases de données ou des bases de connaissances écrites manuellement par des experts du domaine choisi.

Les premiers Chatbots<sup>1</sup> ont également vu le jour peu de temps après comme ELIZA[3], un programme conçu par Joseph Weizenbaum en 1966 au MIT afin d'étudier le langage naturel et la communication entre humains-machines. Il a d'ailleurs réussi à passer le test de Turing<sup>2</sup>.

Un autre programme intéressant nommé SHRDLU[4] qui a fait parler de lui dans le domaine du

1. Interface Homme-Machine de dialogue en langage naturel.  
 2. Test de Turing (voir Wikipédia).

question-answering, a été développé au début des années 70 par Terry Winograd. Le concept était de simuler un environnement virtuel avec des objets en forme de blocs et ainsi pouvoir poser des questions au robot en langage naturel à propos de l'état de l'environnement, des objets etc ...

La plupart des systèmes développés dans le domaine se basait sur les bases de connaissances et systèmes experts et produisait des résultats plutôt satisfaisants dans le cadre de leurs bases de connaissances. Ils restaient tout de même très limités et étaient très dépendants des connaissances des experts du domaine et ne répondaient qu'à des questions bien spécifiques.

C'est dans les années 80 qu'une vague de nouveaux systèmes de QA modernes est apparue. Ces systèmes se basent sur des techniques statistiques et probabilistes capables de traiter de larges corpus non structurés en langage naturel. Ils utilisent des méthodes linguistiques et de compréhension de texte, ce qui a poussé les chercheurs à investir dans la recherche de théories linguistiques et théories du raisonnement pour améliorer ces systèmes.

Des projets ambitieux ont vu le jour comme LILOG d'IBM créé en 1991, un système de compréhension de texte dans le monde du tourisme en Allemagne ainsi que EAGLi dans le domaine de la médecine ou bien Wolfram alpha qui est un système qui répond aux questions en se basant sur des sources depuis le web.

Ce n'est que très récemment avec le développement de l'apprentissage automatique et les réseaux de neurones et la très grande quantité de données générée sur le web que le domaine du question answering a fait un grand pas. En effet, les plus grandes compagnies, chercheurs et universités ont fait de grandes avancées notamment dans la construction de larges corpus et l'implémentation de modèles pouvant même concurrencer la performance humaine dans la compréhension du texte. Avec la quantité faramineuse de données disponibles, les chercheurs se penchent désormais sur le domaine ouvert (open domain), et grâce à cela, les modèles entraînés ne se contentent plus de petites bases de connaissances limitées. Aujourd'hui, ce sont les systèmes qui apprennent le langage naturel à partir des données et apprennent réaliser des tâches spécifiques avec de plus en plus de précision; il n'y a qu'à voir les assistants virtuels *Google Assistant*, *Siri* d'Apple, *Alexa* d'Amazon etc...

Les moteurs de recherches eux aussi se dotent de systèmes de question answering sophistiqués capables de comprendre le langage naturel et de répondre aux questions dans n'importe quel domaine en utilisant ce qu'on appelle des "transformers"<sup>[5]</sup>. Ces derniers sont des modèles de langages puissants capables d'exécuter plusieurs tâches de traitement de langage naturel (NLP). Parmi les modèles les plus connus en ce moment sont *OpenAi*, *Elmo*, *ULMFit*, *Bert* de google<sup>[6]</sup> qui sont tous sortis depuis 2018.

### 1.2.1 Définition et types d'une question :

Une question désigne une phrase qui a pour but de demander ou d'interroger à propos d'un sujet quelconque.

Pour le cas des systèmes de questions-réponses il s'agit d'une requête exprimée en langage naturel pour interroger un système automatique afin d'avoir une réponse relative à cette question.

On peut formuler une question de plusieurs manières et plusieurs formes, il existe donc plusieurs types de questions dont :

**Les questions factuelles :** Une question factuelle est basée sur des faits et peut avoir une réponse en un mot ou en une courte phrase. Les réponses sont généralement des *entités nommées* comme des noms de personnes, des organisations, des dates etc..., C'est le type des "Wh-questions"<sup>3</sup>.

C'est le type de question qui suscite le plus d'intérêt des chercheurs car c'est le plus commun parmi les questions posées par l'homme.

**Exemple 1.2.1** *Exemple :*

*Question : Quelle est la capitale de l'Algérie ?*

*Texte : Ahmed étudie à l'université de Bab Ezzouar à **Alger**, la capitale de l'Algérie.*

**Les questions de choix multiples** La question peut avoir plusieurs choix de réponses comme dans un QCM.

**Exemple 1.2.2** *Exemple :*

*Question : 'De quelle couleur est la voiture de mon père ?'*

*Choix : a) Rouge, b) Bleu, C) Blanche*

*Texte : 'la voiture de mon père est **Bleu**, et celle de ma mère est blanche.'*

**Les questions listes** Ce type de questions a besoin d'une liste de faits ou d'entités comme réponses, par exemple une liste d'ingrédients d'une recette de cuisine dans un texte. Elle est dans la plupart du temps extraite d'articles du web.

**Exemple 1.2.3** *Exemple :*

*Question : 'Quels sont les ingrédients d'une pizza ?'*

*Réponse :*

- Pate.
- Sauce tomate.
- Fromage.
- Olives.
- Poulet

**Les questions booléennes** Les questions booléennes ont des réponses 'oui' ou 'non' , 'vrai' ou 'faux'. Ce type de questions nécessite des mécanismes d'inférence et de raisonnement, et parfois d'autres approches d'apprentissage automatique comme l'analyse de sentiment pour la subjectivité.

---

3. On parle de *Wh-questions* du fait que ces questions sont généralement de la forme : What, Where, Who, Which, Whom, Whose, Why, how, etc.

**Exemple 1.2.4** *Exemple :*

*Question : 'Êtes vous Algérien ? '*

*Réponse : Oui.*

**Les questions causales** Les questions causales cherchent des descriptions, des explications, des causes en relation avec des entités, objets ou événements. La taille de la réponse peut varier d'une simple phrase à un paragraphe voir même jusqu'à un document entier.[\[7\]](#)

**Exemple 1.2.5** *Exemple :*

*Question : 'Pourquoi le ciel est-il bleu ? '*

**Les questions complexes** Les questions complexes sont les plus difficiles à y répondre. Elles contiennent généralement plusieurs conditions et peuvent être composées de plusieurs questions qui requièrent différents types d'informations et différentes sources d'informations et peuvent avoir une ambiguïté qui nécessite de l'inférence. [\[8\]](#)

**Exemple 1.2.6** *Exemples :*

*Où et quand aura lieu le cours de maths ?*

*Quel est le nom de l'acteur qui a joué dans le film qui est sorti en 2019 dans lequel il y a des voitures de courses ?*

## 1.3 Domaine ouvert et Domaine fermé

### 1.3.1 Domaine fermé

Les systèmes QA dans un domaine fermé exploitent une base de connaissances bien spécifique à un domaine comme la médecine, la justice, le sport etc... Ils répondent à un nombre limité de questions qui doivent être du domaine spécifié, mais sont très utiles et performants avec une très haute qualité de réponse pour les personnes du domaine. Les données utilisées sont rarement mises-à-jour et peuvent être structurées (base de données), non structurées (texte libre) ou semi-structurées (XML, documents annotés).

### 1.3.2 Domaine Ouvert

Dans le domaine ouvert, les systèmes QA peuvent traiter pratiquement n'importe quel sujet. Les données sur lesquelles ils travaillent peuvent provenir de n'importe quelle source telles que le web, des bases de données ou des documents. L'avantage du domaine ouvert c'est de pouvoir trouver des données très riches plus facilement que pour un domaine fermé. Cependant, les réponses extraites peuvent être moins précises et plus ambiguës que pour un domaine fermé.

## 1.4 Types de QA systèmes

### 1.4.1 Systèmes basé Texte (Text-Based) :

Ce sont des systèmes qui travaillent essentiellement sur du texte brut ou semi-structuré (HTML, XML...), et il en existe plusieurs sous types comme :

#### i Basé Web (Web-Based) :

Le web est la plus grande source d'information qui existe de nos jours, il contient pratiquement tout type d'informations. Les systèmes QA basés web utilisent un ou plusieurs moteurs de recherche comme : Google, Yahoo, Bing etc.. ainsi que d'autres technologies web variées pour afficher la réponse la plus précise possible des questions posées par l'utilisateur.

Avec la grande quantité disponible de données sur le web, l'utilisateur peut avoir une réponse facilement en utilisant ce type de système. En plus des moteurs de recherche, le système peut utiliser d'autres sources locales ou connectées.

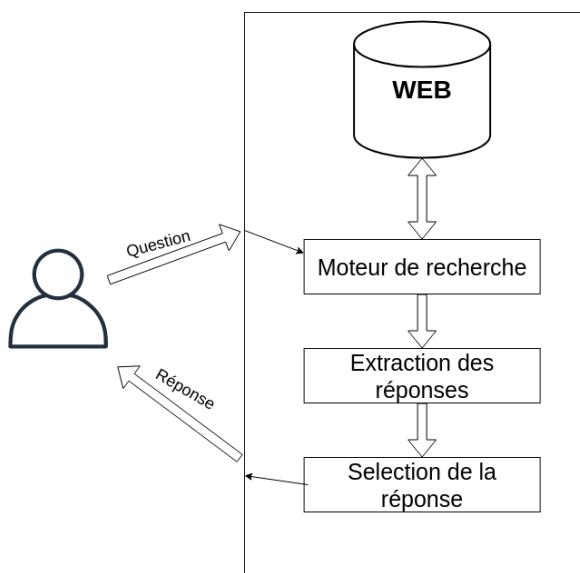


FIGURE 1.2 – Architecture de base d'un système QA basé Web

#### ii Basé Document (Document-based) :

Un système QA basé document est celui qui se rapproche le plus de la compréhension du langage naturel (compréhension du texte / reading comprehension) en utilisant des méthodes du TALN combinées à de l'apprentissage automatique pour arriver à comprendre la sémantique du texte et celle de la question pour donner une réponse adéquate.

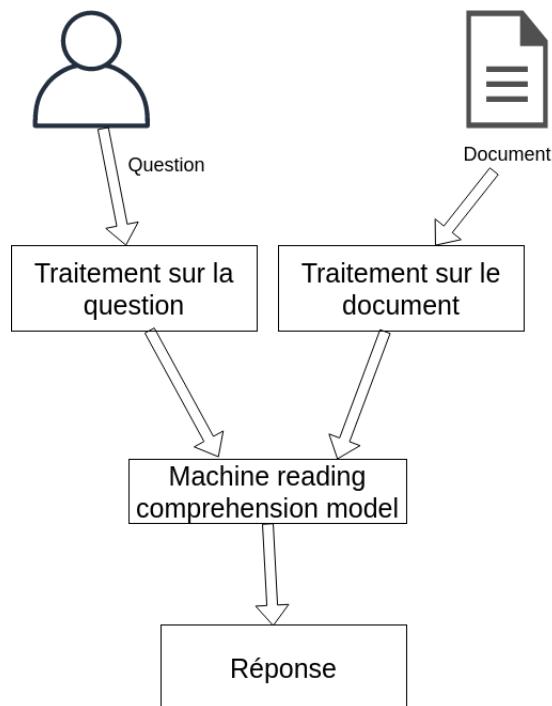


FIGURE 1.3 – Architecture de base d'un système QA basé document

### iii Basé recherche d'informations (IR-Based) :

La recherche d'informations consiste à trouver des résultats (généralement du texte) à partir d'une collection de documents non structurés. Un système basé RI est dans le domaine ouvert et il utilise la collection de documents pour trouver la réponse à la question en utilisant des techniques de recherche d'informations comme les distances et les similarités.

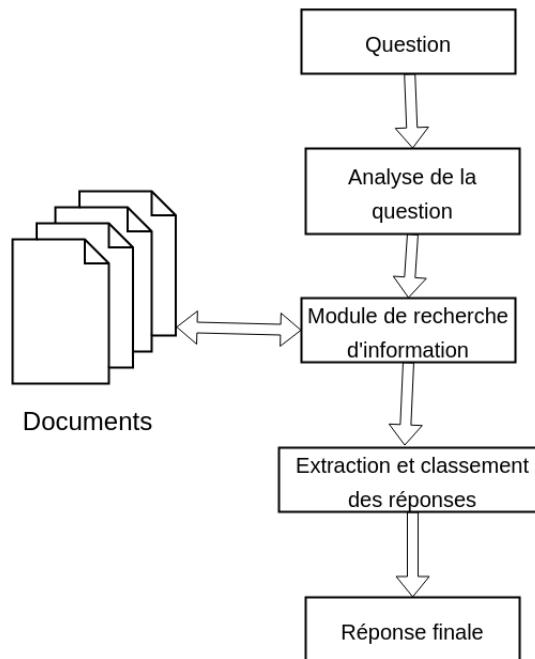


FIGURE 1.4 – Architecture de base d'un système QA basé Recherche d'information

#### 1.4.2 Basé Connaissances (Knowledge-based)

Un système knowledge-based utilise une base de connaissances qui contient des faits. Il reçoit en entrée une question en langage naturel ensuite il envoie une requête qui contient des mots clés ou des entités nommées extraits de la question. Le système vérifie alors les relations entre eux et retourne les données les plus proches par rapport à la question.

La requête est une reformulation de la question exprimée sous forme d'un prédicat. Le moteur d'inférence déduit la réponse à partir de la base des faits.

Il est le plus souvent dans un domaine fermé dans lequel il peut apporter des réponses de qualité.

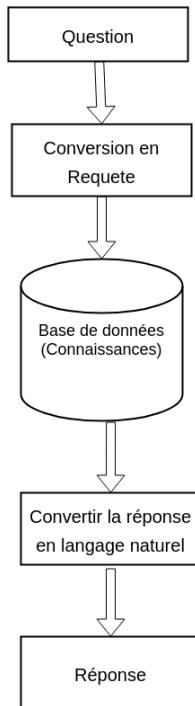


FIGURE 1.5 – Architecture de base d'un système QA basé connaissances

### 1.4.3 Les systèmes hybrides

On peut trouver également des systèmes hybrides qui utilisent plusieurs aspects communs aux autres types de systèmes. Il est même possible de les combiner entre eux, par exemple combiner des systèmes basés IR et Web avec un modèle de compréhension du langage entraîné par apprentissage automatique.

### 1.4.4 Les modèles Deep Learning :

Ce sont des modèles entraînés par apprentissage profond spécialement pour la tâche qui doit extraire une réponse à partir d'un texte donné et une question sur celui-ci. Ces modèles peuvent être incorporés dans n'importe quel autre type de systèmes de questions-réponses.

## 1.5 Les systèmes fréquents d'application du question answering :

- FAQ (Foire aux questions) : Les utilisateurs d'un produit ou service quelconque ont tendances à chercher des réponses à leurs questions concernant une information ou un problème à propos du produit. L'utilisateur peut passer beaucoup de temps à chercher manuellement une réponse à sa question dans une foire aux questions classique. Pour cela, des systèmes de questions-réponses peuvent être mis en place pour répondre directement à la question de l'utilisateur en se basant sur des connaissances à propos du produit.[9]
- Assistant virtuel : La plupart des Smartphones récents ont un assistant virtuel qui est une application qui permet d'aider la personne dans différentes tâches, comme répondre à des

messages, ouvrir une application, prendre une photo, ou répondre à des questions etc ... Il en existe plusieurs systèmes de ce type, comme Siri d'Apple, Alexa d'Amazon ou google assistant de GOOGLE.



FIGURE 1.6 – Exemple d'utilisation de google assistant

- Moteur de recherche : Les moteurs de recherche sont aussi depuis peu dotés d'un système de questions-réponses beaucoup plus précis qui peut retourner une réponse exacte à l'utilisateur sans le laisser passer par plusieurs liens d'articles, ce qui permet un gain de temps impressionnant.



FIGURE 1.7 – Exemple de réponse QA de google

## 1.6 L'apprentissage profond

L'apprentissage profond (Deep Learning) est un sous domaine de l'apprentissage automatique qui vise une plus grande précision et différentes manières de traiter les données et d'en extraire des caractéristiques. Les réseaux d'apprentissage profond sont capables d'apprendre depuis des données non structurées et non étiquetées. Il peut supporter une très quantité de données grâce aux différentes architectures plus complexes et robustes.

Le Deep Learning a donné des résultats intéressants pour les différents problèmes du TALN, telles que la :

- Compréhension du langage.
- Traduction automatique.
- Génération de texte.
- Analyse de sentiments.
- Reconnaissance d'entités nommées.

### 1.6.1 Word embeddings

Word embeddings est une approche d'apprentissage profond qui permet de représenter des mots par des vecteurs de telle manière que les mots apparaissant dans des contextes similaires ont des représentations voisines. Il existe deux principales méthodes pour représenter des mots en vecteurs.

La première méthode appelée « Continuous Bag of Words » (CBOW), qui entraîne le réseau de neurones pour prédire un mot en fonction de son contexte, c'est-à-dire les mots avant/après dans une phrase. Dans la seconde méthode, on essaie de prédire le contexte en fonction du mot. C'est la technique du « skip-gram ». Et cela signifie que les mots soient caractérisés par les mots qui les entourent le plus.

Dans les deux cas, le réseau de neurones comporte deux couches. La couche cachée contient quelques centaines de neurones et constitue, à l'issue de la représentation, le prolongement lexical (embedding) permettant de représenter un mot. La couche de sortie permet d'implémenter une tâche de classification.

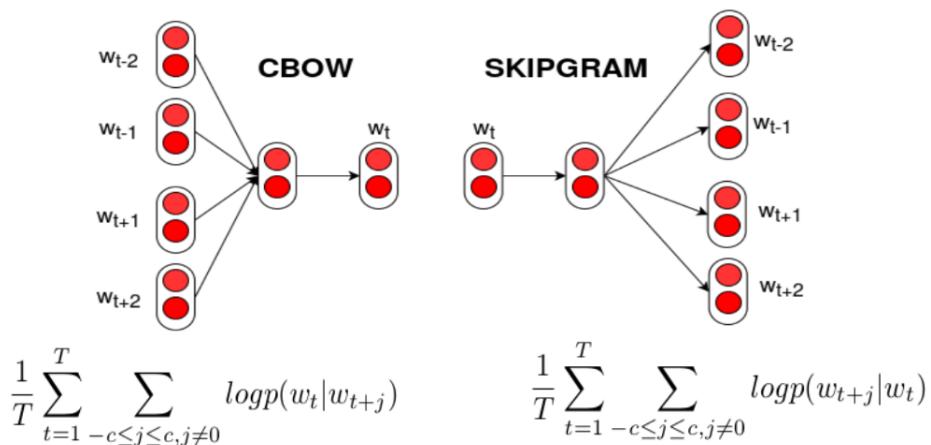


FIGURE 1.8 – Architectures de CBOW et Skipgram

Dans ce projet, nous allons utiliser un modèle pré-entraîné qui permet de représenter les mots en vecteurs tout en gardant leurs aspect sémantique, c'est-à-dire que si il y a des mots qui sont similaires ils vont forcément avoir des représentations assez proches l'une de l'autre. Ce modèle se

nomme GLOVE[10] et nous allons nous intéresser au processus par lequel il passe afin d'accomplir cette tâche.

Glove vise à combiner la factorisation matricielle basée sur la co-occurrence et le modèle skip-gram basé sur le contexte. Comme le sens d'un mot est lié aux mots qui l'entourent, ce modèle utilise une matrice de co-occurrence pour calculer le nombre d'apparitions et la probabilité d'un mot dans le contexte de chaque autre mot.

#### Exemple 1.6.1 (Formule de co-occurrence )

$$P_{co}(w_k|w_i) = \frac{C(w_i, w_k)}{C(w_i)}$$

$C(w_i, w_k)$  compte le nombre d'apparitions entre le mot  $w_i$  et le mot  $w_k$ .

Le modèle a été entrainé avec une combinaison de skipgram et de matrice de co-occurrence sur un vocabulaire de plus de 400 000 mots. Pour comprendre ce qui se passe, prenons un exemple :

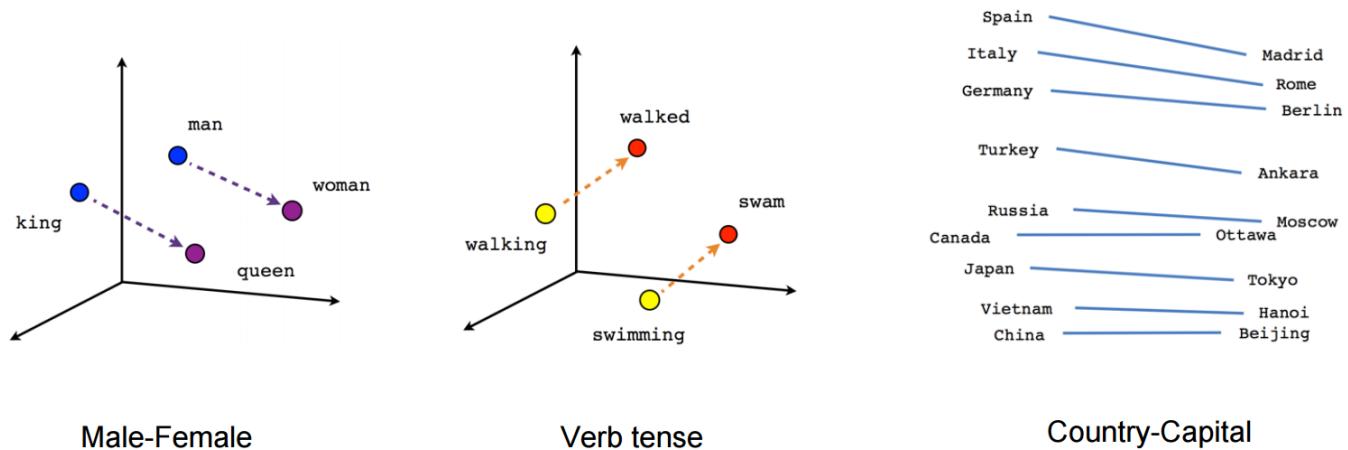


FIGURE 1.9 – Exemple des relations entre les mots en utilisant Glove.

Source: [openclassrooms.com](https://www.openclassrooms.com)

Comme nous pouvons le voir ici, après la représentation des mots en vecteurs et leurs visualisation, on peut voir que les mots " Man " et " Woman " sont très proches, "Italy" et "Rome" etc ...

Autre exemple :

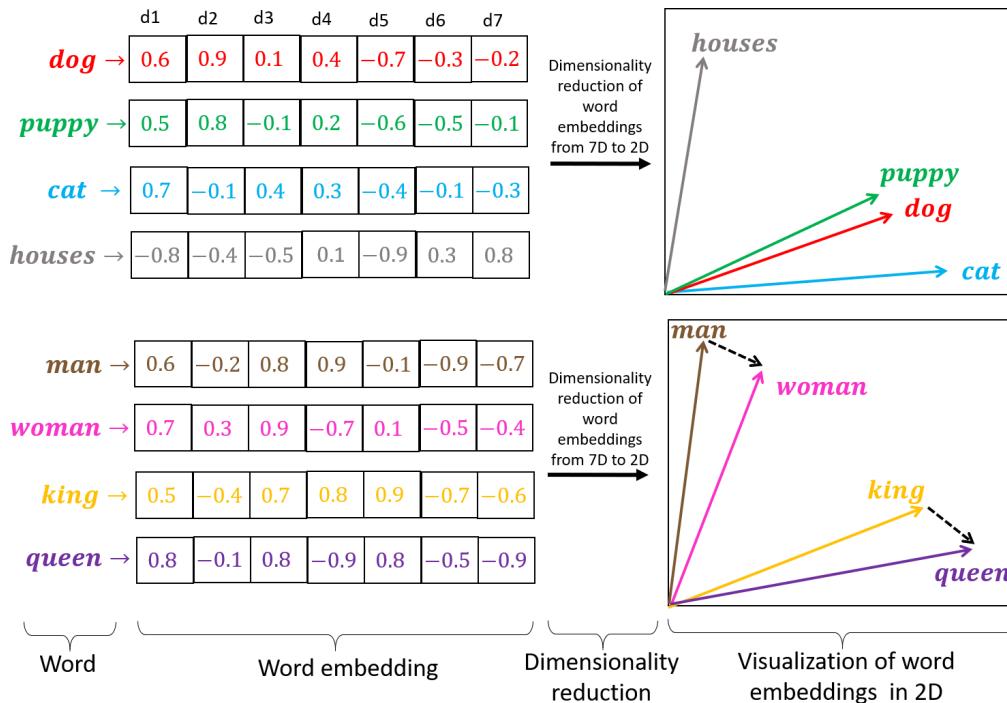


FIGURE 1.10 – Exemples de représentations avec Glove

Source: [towardsdatascience.com](http://towardsdatascience.com)

L'utilisation des Word Embedding en tant que représentation du texte garantit plusieurs avantages :

- la représentation de la sémantique des mots et les relations entre eux, donc une représentation plus significative des mots pour la machine;
- l'optimisation de la mémoire par la réduction des tailles des vecteurs de représentation;
- la précision améliorée pour l'utilisation des réseaux de neurones.

## 1.6.2

### i Réseaux de neurones récurrents (RNN)

Un réseau de neurones récurrents est un réseau de neurones artificiels présentant des connexions récurrentes. Il contient des boucles, permettant de stocker des informations dans le réseau. Son raisonnement est basé sur des expériences précédentes pour calculer les expériences à venir.

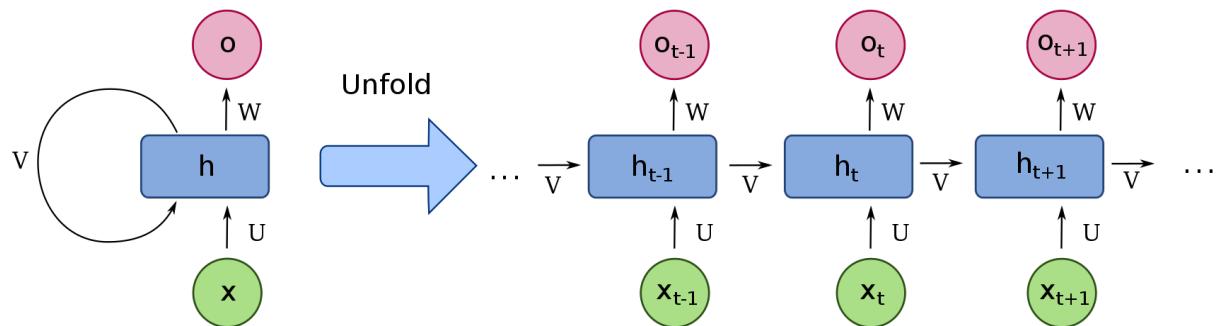


FIGURE 1.11 – Schéma basique d'une architecture de réseaux de neurones récurrents.

**Source:** wikipedia.org

Les RNN sont composés d'une couche d'entrée et une de sortie de taille égale à la taille de la séquence d'entrée et de sortie, ainsi qu'une couche cachée qui lie séquentiellement les états cachés. Ils utilisent les précédents états cachés et les entrées correspondantes pour prédire les prochains états cachés ainsi que leurs sorties.[11]

$$\text{Hidden State } h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

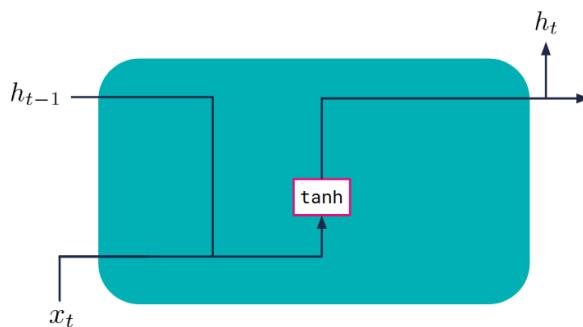


FIGURE 1.12 – Schéma d'une cellule d'un réseaux de neurones récurrents

**Source:** Andrej Karpathy blog, karpathy.github.io

### Variables et fonctions

- $x_t$  : Vecteur d'entrée.
- $h_t$  : Vecteur de la couche cachée.
- $o_t$  : Vecteur de sortie.
- $W, U$  et  $b$  : Matrices et vecteur (paramètres).
- $\tanh$  Tangente hyperbolique : fonction d'activation

Les réseaux de neurones récurrents peuvent avoir plusieurs formes pour faire plusieurs tâches différentes du traitement de langage naturel, voici un schéma de quelques-unes :

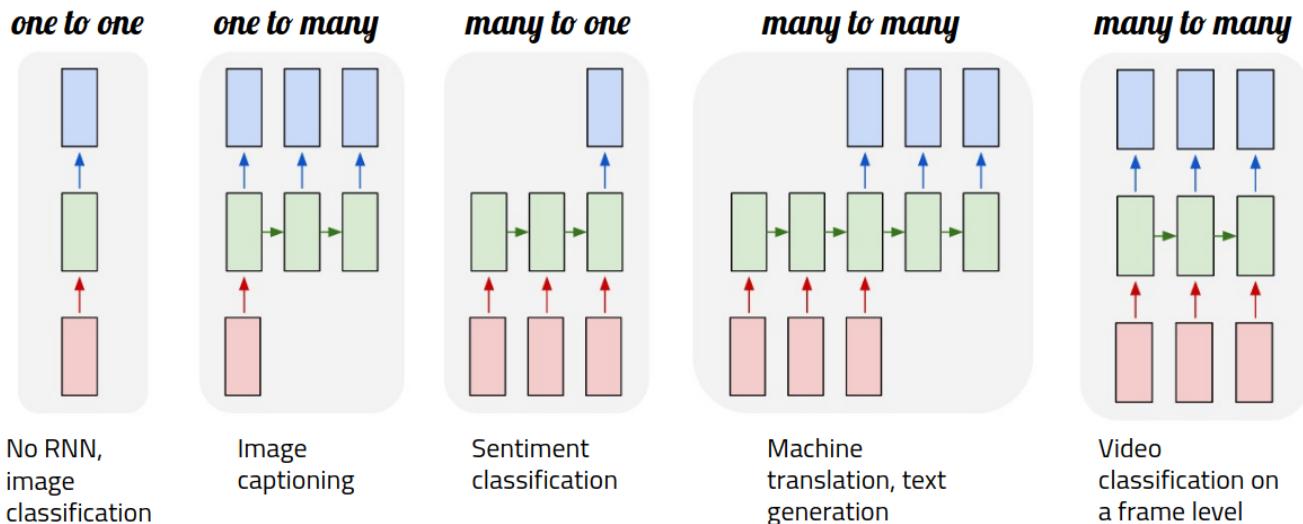


FIGURE 1.13 – Types d'utilisation de réseaux de neurones récurrents  
Andrej Karpathy blog, [karpathy.github.io](http://karpathy.github.io)

Ces modèles sont précieux dans leur capacité à traiter des séquences de données. Dans le cas du TALN, une phrase est une séquence de mots, et le sens de ces mots dépend énormément des autres qui les précédent, alors il serait judicieux de les mémoriser pour avoir encore plus de précision dans les prochaines prédictions. Néanmoins, en pratique ces réseaux de neurones récurrents peuvent être assez limités en traitant de longues séquences à cause du problème de la disparition du gradient.

## ii Le problème de la disparition du gradient dans les RNN

À mesure que des couches supplémentaires sont ajoutées aux réseaux neurones, qui utilisent certaines fonctions d'activations, les gradients de la fonction de perte s'approchent de zéro. Ceci qui rend le réseau difficile à former, car certaines fonctions d'activation, comme la sigmoïde, compriment un grand espace d'entrée dans un petit espace d'entrée compris entre 0 et 1. Par conséquent, une grande modification de l'entrée de la fonction sigmoïde entraînera une petite modification de la sortie. Par conséquent, la dérivée devient petite.

Lorsque les entrées de la fonction sigmoïde deviennent plus ou moins grandes (lorsque  $|x|$  devient plus grand<sup>4</sup>), la dérivée devient proche de zéro. Pour un réseau peu profond avec seulement quelques couches qui utilisent ces activations, ce n'est pas un gros problème. Cependant, lorsque plusieurs couches sont utilisées, la pente peut être trop faible pour que la formation fonctionne efficacement.

4. X étant l'entrée, la valeur qu'on donne au neurone.

Les gradients des réseaux de neurones sont calculés en utilisant la rétropropagation. La rétropropagation permet de trouver les dérivés du réseau en se déplaçant couche par couche de la couche finale à la couche initiale. Les dérivés de chaque couche sont multipliés le long du réseau, de la couche finale à la couche initiale. Cependant, lorsque  $n$  couches cachées utilisent une activation comme la fonction sigmoïde,  $n$  petites dérivées sont multipliées ensemble. Ainsi, le gradient diminue de façon exponentielle au fur et à mesure que l'on se propage vers les couches initiales. Un petit gradient signifie que les poids et les biais des couches initiales ne seront pas mis à jour efficacement à chaque séance d'entraînement. Comme ces couches initiales sont souvent cruciales pour reconnaître les éléments essentiels des données d'entrée, cela peut conduire à une imprécision globale de l'ensemble du réseau.

Pour résoudre ce problème, on peut utiliser d'autres fonctions d'activation, comme la fonction ReLU, qui ne provoque pas de petites dérivées. Mais on peut aussi utiliser d'autres formes de réseaux récurrents qui sont créés pour régler ce problème, comme les LSTM (Long short-term memory) et GRU (Gated Recurrent Unit).

### iii Long short-term memory (LSTM) et Gated Recurrent Unit (GRU)

Les réseaux long short-term memory (LSTM) et gated recurrent unit (GRU) qui sont également des réseaux de neurones récurrents ont été créés comme solution pour le problème de perte de mémoire (Disparition du gradient). La différence avec les RNN est l'ajout de mécanismes de portes de contrôle qui peuvent réguler le flux d'information. Ces portes de contrôle peuvent apprendre quelle information est importante à garder ou au contraire à oublier. Ces types d'architectures ont déjà prouvé leur efficacité pour presque tous les aspects du traitement automatique du langage (TALN).

**LSTM** Les réseaux de neurones de type LSTM fonctionnent de la manière suivante :

**Cell State :** Un chemin de transport par lequel les informations importantes passent, on peut le considérer comme la mémoire du réseaux. Des informations peuvent être ajoutées ou supprimées avant d'arriver à celui ci.

**Forget Gate :** Cette porte décide si une information qui vient des étapes précédentes est importante ou non, si elle doit être gardée ou oubliée. Le précédent état caché et l'entrée en cours passent à travers une fonction sigmoïde (voir annexe) pour avoir en sortie une valeur entre 0 et 1, si la valeur est proche de 0 alors l'information est oubliée, sinon si elle est proche du 1 l'information est gardée.

**Input Gate :** Pour mettre à jour l'information qui passe par Cell state, le précédent état caché et l'entrée en cours passent par une fonction sigmoïde et une autre fonction Tangente hyperbolique puis les deux résultats sont alors multipliés. Cette porte décide de ce qui est important dans l'information en cours.

**Output Gate :** Cette porte décide de comment doit être le prochain état caché. Il est calculé par la combinaison de l'information du Cell State qui passe par une fonction sigmoïde avec l'état caché précédent combiné à l'entrée en cours et qui passent eux aussi par une même fonction.

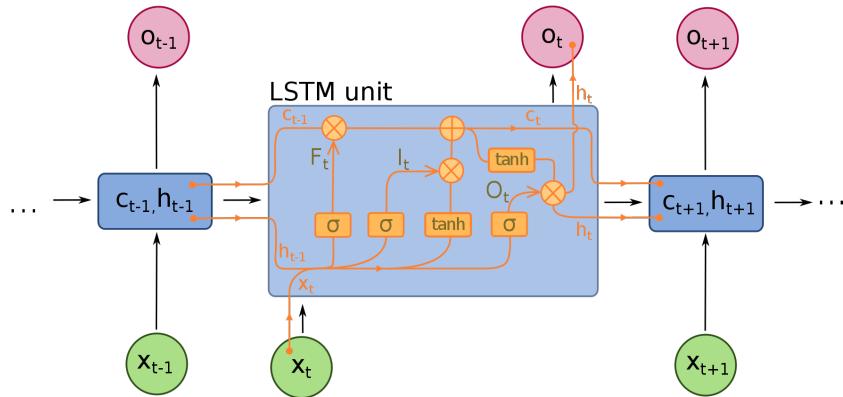


FIGURE 1.14 – Schéma d'une cellule LSTM  
Source : Wikipedia

#### Exemple 1.6.2 (Formules du LSTM :)

$$F_t = \sigma(W_F x_t + U_F h_{t-1} + b_F) \quad (forget\ gate) \quad (1.1)$$

$$I_t = \sigma(W_I x_t + U_I h_{t-1} + b_I) \quad (input\ gate) \quad (1.2)$$

$$O_t = \sigma(W_O x_t + U_O h_{t-1} + b_O) \quad (output\ gate) \quad (1.3)$$

$$c_t = F_t \circ c_{t-1} + I_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (1.4)$$

$$h_t = O_t \circ \tanh(c_t) \quad (1.5)$$

$$o_t = f(W_o h_t + b_o) \quad (1.6)$$

**GRU** Les réseaux de neurones de type Gated Recurrent Unit (GRU) sont une version récente de RNN et très similaires aux LSTM, mais n'ont que les deux portes suivantes :

- Update Gate : Similaire aux portes Forget Gate et Input Gate des LSTM. Elle décide quelles informations garder ou au contraire oublier.
- Reset Gate : Elle décide de la quantité d'informations précédentes à oublier.

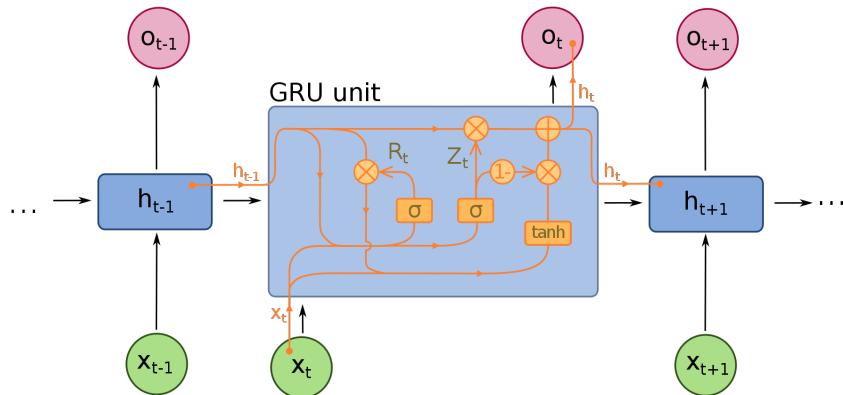


FIGURE 1.15 – Schéma d'une cellule GRU  
Source : Wikipedia

### Exemple 1.6.3 (Formules de GRU :)

$$Z_t = \sigma(W_Z x_t + U_Z h_{t-1} + b_Z) \quad (update\ gate) \quad (1.7)$$

$$R_t = \sigma(W_R x_t + U_R h_{t-1} + b_R) \quad (reset\ gate) \quad (1.8)$$

$$h_t = Z_t \circ h_{t-1} + (1 - Z_t) \circ \tanh(W_h x_t + U_h (R_t \circ h_{t-1}) + b_h) \quad (1.9)$$

Pour résumer, les RNN sont appropriés pour le traitement des données de séquence pour les prédictions mais souffrent de mémoire à court terme. Les LSTM et GRU ont été créés comme méthode pour atténuer la mémoire à court terme en utilisant des mécanismes appelés portes. Les portes ne sont que des réseaux de neurones qui régulent le flux d'informations circulant dans la chaîne de séquence.

Les LSTM et GRU sont utilisés dans des applications d'apprentissage profond de pointe telles que la reconnaissance vocale, la synthèse vocale, la compréhension du langage naturel, etc..

### 1.6.3 Modèle de séquence à séquence (Seq2Seq model)

Le modèle de séquence à séquence (Seq2Seq) est un modèle d'apprentissage profond souvent utilisé dans des tâches comme la traduction automatique, le résumé de texte et le sous-titrage d'images. Il prend une séquence d'éléments en entrée et produit une autre séquence d'éléments en sortie.

Le modèle est composé d'un encodeur et d'un décodeur. L'encodeur capture le contexte de la séquence d'entrée sous la forme d'un vecteur de l'état caché et l'envoie au décodeur, qui produit alors la séquence de sortie selon celui-ci. L'encodeur et le décodeur ont tendance à utiliser une certaine forme de RNN, LSTM, GRU, etc.

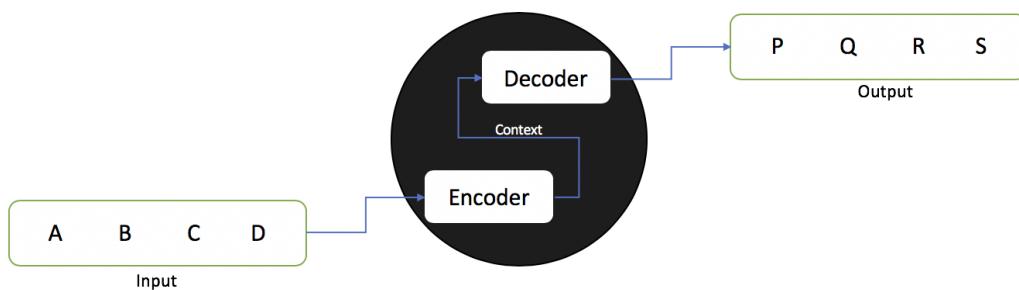


FIGURE 1.16 – Schéma simplifié d'un modèle de séquence à séquence (seq2seq)

Source: [towardsdatascience.com](https://towardsdatascience.com/introduction-to-seq2seq-models-11f3a2a3a3d)

L'encodeur prend la séquence en entrée et concatène tous les vecteurs de l'état caché des éléments de la séquence pour générer un seul vecteur à la fin de la séquence appelé aussi un vecteur contexte. Celui-ci est ensuite envoyé au décodeur qui l'utilise pour prédire une séquence [12].

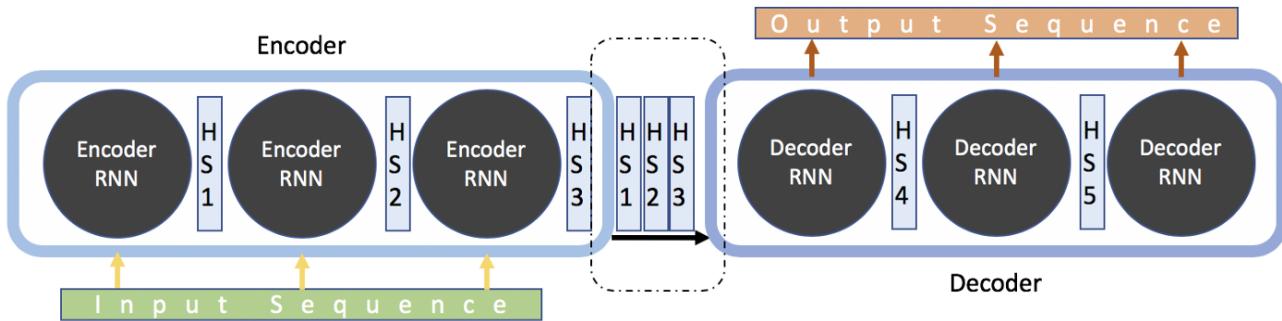


FIGURE 1.17 – Schéma d'un modèle de séquence à séquence (seq2seq)

Source: [towardsdatascience .com](https://towardsdatascience.com/)

#### Exemple 1.6.4 (Formules du modèle :)

##### Encodeur :

$$h_t^e = f(h_{t-1}^e \cdot h_t^e) \quad (1.10)$$

Pour chaque entrée, son vecteur de l'état caché est concaténé avec son précédent.

##### Décodeur :

$$h_t^d = g(h_{t-1}^e \cdot h_{t-1}^d) \quad (1.11)$$

$$Z_t = f(h_t^d) \quad (1.12)$$

$$Y_t = \text{softmax}(Z_t) \quad (1.13)$$

Pour chaque prédiction d'un mot, le décodeur prend en entré le précédent vecteur d'état caché  $h_{t-1}^d$  ainsi que celui de l'encodeur  $h_{t-1}^e$  puis le résultat  $h_t^d$  passe par la couche RNN qui lui associée pour enfin passer par une fonction softmax pour prédire le mot résultant  $Y_t$ .

Voici un exemple d'un modèle de séquence à séquence dans le domaine de la traduction automatique :

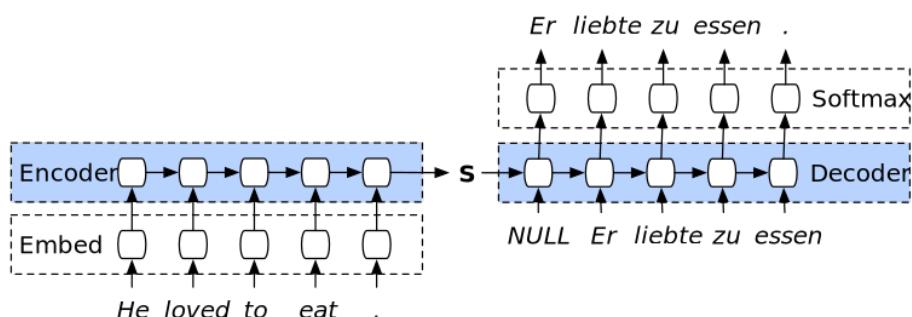


FIGURE 1.18 – Exemple d'un modèle seq2seq en traduction automatique

Source: [smerity .com](https://smerity.com)

### 1.6.4 Mécanisme d'attention

Dans un modèle de séquence à séquence, le résultat final dépend fortement du contexte défini par l'état caché dans la sortie finale de l'encodeur, ce qui rend difficile pour le modèle de traiter les longues phrases car le vecteur contexte a une taille fixe. Dans le cas de longues séquences, il y a une forte probabilité que le contexte initial ait été perdu à la fin de la séquence. Pour remédier à ce problème, un mécanisme a été créé sous le nom de "Mécanisme d'attention" qui permet au modèle de se concentrer sur différentes parties de la séquence d'entrée à chaque étape de la séquence de sortie.

L'idée principale derrière ce mécanisme est de rediriger l'attention du modèle pendant la prédiction sur les mots les plus importants selon la tâche. Ceci lui permet d'être plus précis et de ne pas perdre de temps avec les mots qui ne sont pas importants durant le processus.

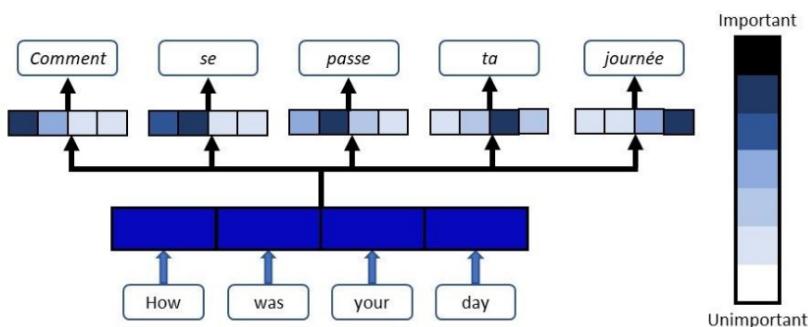


FIGURE 1.19 – Exemple d'attention en traduction automatique

Source: [/blog.floydhub.com](http://blog.floydhub.com)

Comme on peut le voir, pour chaque mot de la phrase en français, nous avons l'importance de chaque mot de la phrase en anglais.

Le mécanisme d'attention est appliqué de cette façon :

- Produire un vecteur de scores qui capture l'importance de chaque état de l'encodeur par rapport à l'état en cours du décodeur. Pour cela, le produit scalaire des deux vecteurs est calculé.

$$\text{Score}(h_{i-1}^d, h_j^e) = h_{i-1}^e \cdot h_{t-1}^d \quad (1.14)$$

Le résultat est un produit scalaire entre les deux vecteurs d'états cachés de l'encodeur avec le décodeur en cours qui mesure la similarité entre les deux. Mais ce vecteur reste un vecteur statique qui ne s'adapte pas facilement avec les caractéristiques du processus. Une technique plus robuste a été introduite. Elle consiste à ajouter un vecteur de poids paramétrables  $W_s$  pour donner un aspect d'apprentissage et améliorer la précision de la similarité entre les vecteurs.[12]

$$\text{Score}(h_{i-1}^d, h_j^e) = h_{i-1}^e \cdot W_s h_{t-1}^d \quad (1.15)$$

Une fonction *softmax* est appliquée sur le vecteur de score pour le normaliser et donner une distribution de probabilité pour chaque mot du décodeur par rapport à chaque mot du décodeur.

$$C_i = \text{softmax}(\text{Score}(h_{i-1}^d, h_j^e)) \quad (1.16)$$

Maintenant que le vecteur d'attention est prêt, il peut être ajouté à l'entrée de chaque mot du décodeur avec les entrées précédemment expliqués dans le modèle seq2seq.

$$h_t^d = g(h_{t-1}^e \cdot h_{t-1}^d \cdot C_t) \quad (1.17)$$

Le mécanisme d'attention est devenu très important pour un modèle qui traite des séquences et produit des séquences. Il permet notamment de se concentrer sur la partie la plus importante d'une séquence pour ensuite mieux prédire le résultat avec plus de précision. Il est notamment aussi utilisé dans le domaine de la vision artificielle pour se concentrer sur une partie importante d'une image afin de faire sa prédiction.

### 1.6.5 Architecture Transformer :

Le document «Attention Is All You Need»[13] datant de 2017 présente une nouvelle architecture appelée Transformer. Comme le titre l'indique, il utilise le mécanisme d'attention vu précédemment. Comme les LSTM et GRU, Transformer est une architecture permettant de transformer une séquence en une autre à l'aide de deux parties (encodeur et décodeur), mais il diffère des modèles séquence à séquence précédemment décrits / existants car il n'implique aucun réseau récurrent (GRU, LSTM, etc.).

Les réseaux récurrents étaient, jusqu'à présent, l'un des meilleurs moyens de capturer les contextes des séquences. Cependant, l'équipe de présentation de l'article a prouvé qu'une architecture avec uniquement des mécanismes d'attention sans RNN (réseaux neuronaux récurrents) peut améliorer les résultats dans plusieurs tâches du TALN.

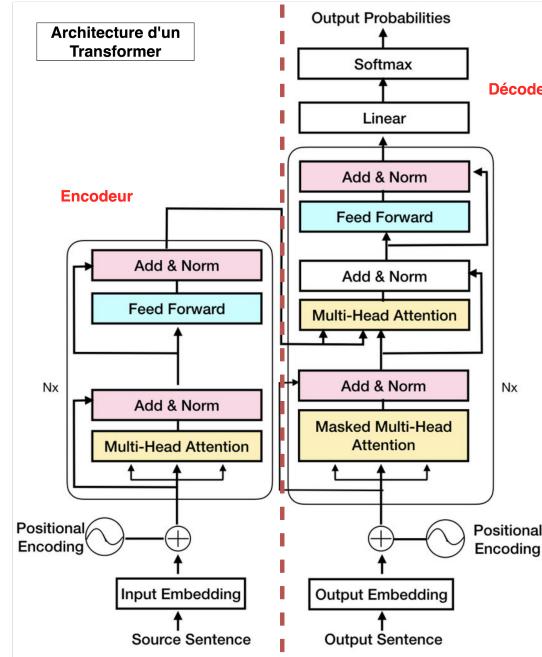


FIGURE 1.20 – Architecture de base d'un modèle Transformer

Ce type d'architecture utilise plusieurs couches de mécanisme d'attention qu'ils appellent "Multi-head Attention" dans l'encodeur et décodeur en même temps.

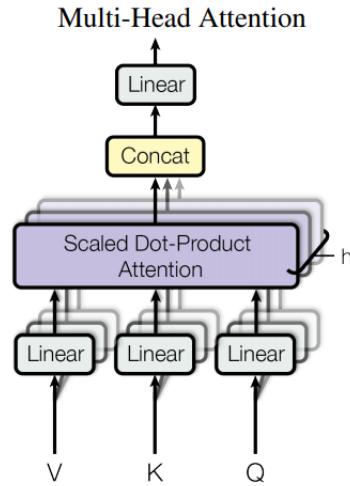


FIGURE 1.21 – Schéma représentant une multicouche d'attention d'un Transformer

L'encodeur et le décodeur sont composés de modules qui peuvent être empilés les uns sur les autres plusieurs fois, ce qui est décrit par Nx dans la figure. Nous voyons que les modules sont principalement constitués de couches Multi-Head Attention et Feed Forward. Une partie légère

mais importante du modèle est le codage positionnel des différents mots. Puisque nous n'avons pas de réseaux récurrents qui peuvent se souvenir de la façon dont les séquences sont introduites dans un modèle, nous devons d'une manière ou d'une autre donner à chaque mot de la séquence une position relative puisqu'une séquence dépend de l'ordre de ses éléments. Ces positions sont ajoutées à la représentation intégrée (vecteur à  $n$  dimensions) de chaque mot.

Le plus important dans cette architecture est que le mécanisme a en entrée un vecteur de représentation d'un mot  $Q$ , des vecteurs de représentation de tous les mots  $K$ , et  $V$  leurs valeurs. Le principe consiste à appliquer pratiquement la même formule d'attention expliquée précédemment en plusieurs fois selon le nombre de couches d'attention.

#### **Exemple 1.6.5 (Formule d'attention d'une architecture Transformer :)**

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q * K^T}{\sqrt{d_k}}\right)V$$

$Q$  : Vecteur de représentation d'un mot

$K$  : Vecteur de représentation de tous les mots.

$V$  : Valeurs des mots

$d_k$  : dimension de toute la séquence

Cette architecture est très importante pour comprendre la suite de la conception du modèle par apprentissage par transfert.

#### **1.6.6 L'apprentissage par transfert**

L'apprentissage par transfert est l'un des champs de recherche de l'apprentissage automatique qui vise à transférer des connaissances d'une ou de plusieurs tâches sources vers une ou plusieurs tâches cibles.

Les connaissances apprises d'un modèle pour résoudre un problème donné peuvent être transférées vers un nouveau modèle pour apprendre une autre tâche similaire dans un autre domaine. Par exemple, un modèle peut apprendre à reconnaître des camions à partir des connaissances d'un modèle qui reconnaît des voitures.

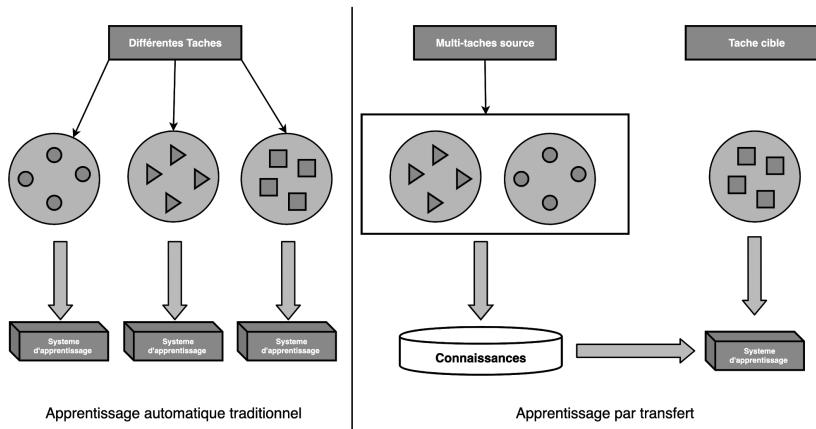


FIGURE 1.22 – Comparaison de l'apprentissage traditionnel avec l'apprentissage par transfert

Cette technique permet entre autre de ne pas entraîner un modèle de zéro et de profiter des connaissances déjà acquises de modèles déjà performants dans certains domaines pour résoudre des problèmes dans d'autres domaines tout en utilisant une quantité de données beaucoup moins importante que dans un apprentissage traditionnel.

Il existe différents types d'apprentissage par transfert, mais dans cette approche nous allons utiliser le réglage fin d'un modèle (*Fine-tuning*). Il consiste à prendre un modèle pré-entraîné dans un domaine et d'y enlever la dernière couche de prédiction et d'ajouter une nouvelle couche adaptée à la nouvelle tâche que le modèle doit apprendre tout en laissant le reste des modèles couches pouvant être entraînées en modifiant les poids de leurs neurones.

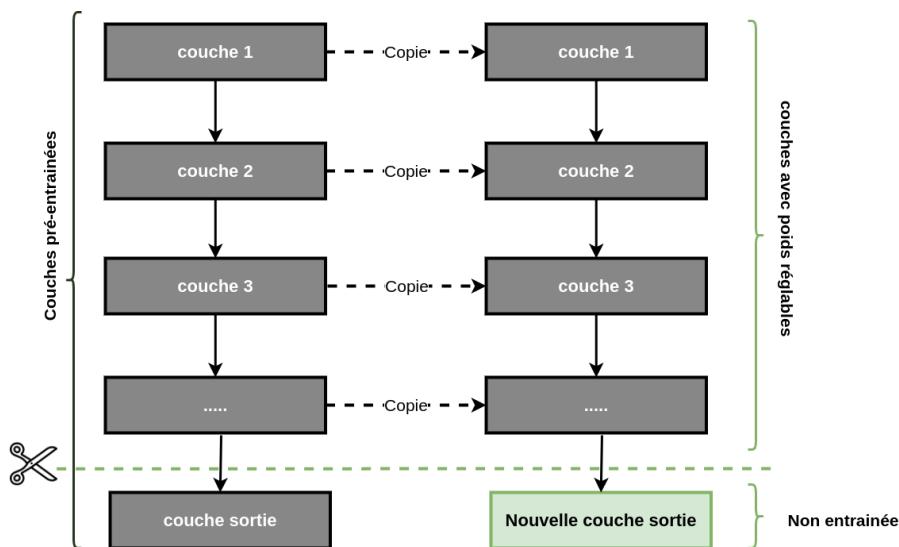


FIGURE 1.23 – Principe de base du réglage fin (Fine-tuning)

Nous avons présenté le Question Answering, sa définition et son historique ainsi que les notions du Deep Learning nécessaires à la compréhension de ce travail, dans le chapitre suivant nous expliquons les solutions que nous proposons à ce problème.

# CHAPITRE 2

## CONCEPTION

Motivés par les succès du Deep Learning dans les différents tâches du TALN et surtout dans la tâche de la modélisation du langage, nous avons conçu nos solutions à base des techniques de Deep Learning. Dans ce chapitre nous expliquons nos solutions avec les détails de chacune.

### 2.1 Stanford Question Answering Dataset (SQuAD)

#### 2.1.1 SQuAD : Un dataset de questions factuelles

SQuAD est un corpus de compréhension écrite construit par une équipe de chercheurs en TALN de l'université Stanford[14]. Il est constitué de questions factuelles posées sur des articles de Wikipédia en anglais. Les réponses sont des passages qui se trouvent dans les articles.

Pour le construire, ses auteurs ont d'abord utilisé le projet Wikipedia's internal PageRank[15] qui consiste à extraire les articles Wikipedia les plus pertinents, ils ont extrait les 10 000 premiers puis n'ont gardé que 536 articles. Ils ont ensuite filtré le contenu pour ne garder que les paragraphes de l'article. Par la suite, ils ont embauché des travailleurs pour créer des questions et y répondre pour chacun des paragraphes de chaque article.

##### i Quelques détails sur le dataset

- Date de publication : 16 juin 2016
- Taille : ~ 100 000 (questions/réponses), 107785 exactement
- 23,215 paragraphes , 536 articles.
- Domaine ouvert et Questions Factoïdes.
- Fichier JSON de structure :

```

Article >>
Texte >>
Questions >>
(Question,Réponse texte, debut réponse,fin réponse).

```

### **Extrait du dataset**

**Texte :**

Established originally by the Massachusetts legislature and soon thereafter named for [John Harvard](#) (its first benefactor), Harvard is the United States' oldest institution of higher learning, and the Harvard Corporation (formally, the President and Fellows of Harvard College) is its first chartered corporation. Although never formally affiliated with any denomination, the early College primarily trained Congregationalist and Unitarian clergy. Its curriculum and student body were gradually secularized during the 18th century, and by the 19th century Harvard had emerged as the central cultural establishment among Boston elites. Following the American Civil War, President Charles W. Eliot's long tenure (1869–1909) transformed the college and affiliated professional schools into a modern research university; Harvard was a founding member of the Association of American Universities in 1900. James Bryant Conant led the university through the Great Depression and World War II and began to reform the curriculum and liberalize admissions after the war. The undergraduate college became co-educational after its 1977 merger with Radcliffe College.

**Question :**

What individual is the school named after?

**Réponse :**

John Harvard

**Debut de la réponse :** 11

**Fin de la réponse :** 12

### **ii Quelques statistiques descriptives sur le dataset**

D'après les statistiques effectuées sur le dataset (Voir Figure ii), plusieurs remarques peuvent être signalées :

- La taille maximale des textes est de 435 mots et la moyenne de ~ 150 mots.
- La taille maximale des questions est de 32 mots et la moyenne de ~ 12 mots.
- La taille maximale des réponses est de 26 mots et la moyenne de ~ 2-3 mots.

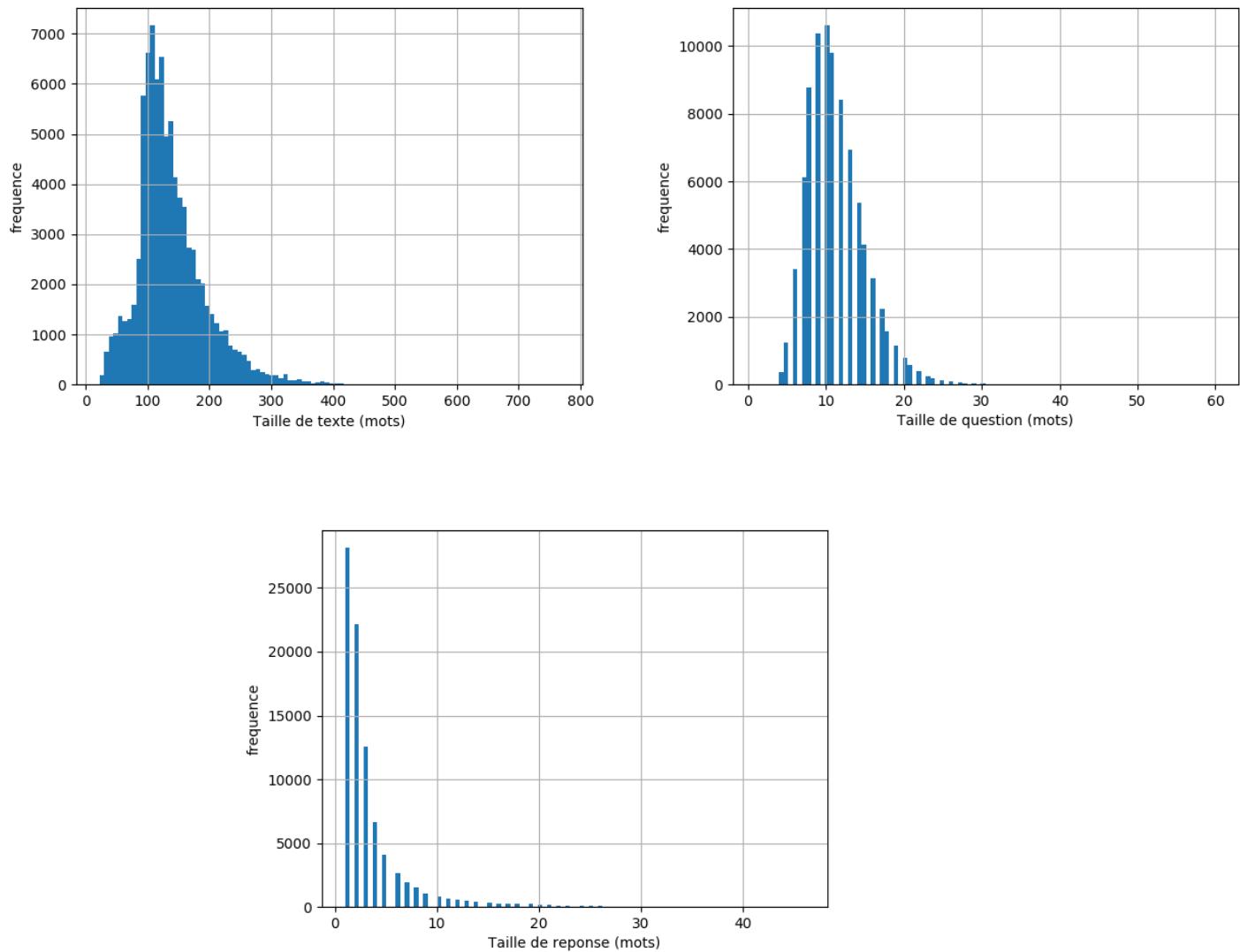


FIGURE 2.1 – Fréquences des textes par rapport à leur taille

Afin de concevoir et de tester nos solutions, nous avons choisi de répartir le dataset entre les tâches de l'entraînement, la validation et les tests de la manière suivante :

TABLE 2.1 – Répartition du corpus pour les tâches de l'entraînement, validation et test.

Taille du corpus	Nombre de paires (texte,question)	Pourcentage
Corpus total	102324	100%
Corpus d'entraînement + validation	87599	85%
Corpus d'entraînement	74459	70%
Corpus de validation	13140	15%
Corpus de test	14725	15%

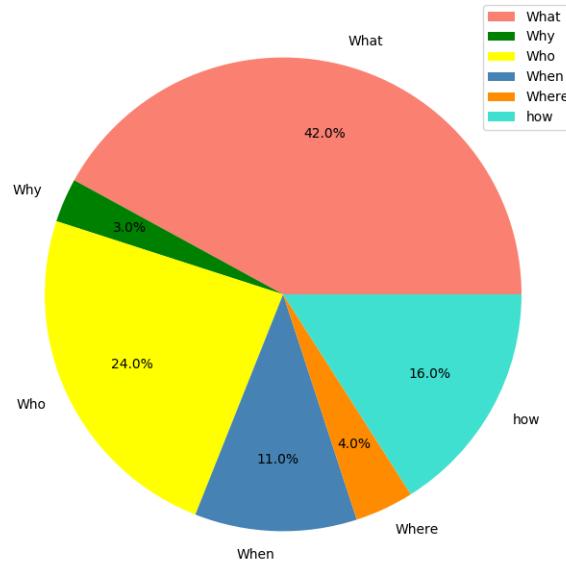


FIGURE 2.2 – Distributions des *Wh-questions* dans le dataset SQuAD.

### 2.1.2 Prétraitements sur le dataset

#### i Nettoyage du dataset

**Elimination des accents et remplacement des caractères spéciaux :** Nous avons d'abord commencé par éliminer les accents qui pourraient être dans des noms de personnes, lieux etc...

Exemple :

On March 17, 1752, the Governor-General of New France, Marquis de la Jonquière, died and was temporarily replaced by Charles le Moyne de Longueuil.

Jonquière »»» Jonquiere

D'autres remplacements moins évidents aussi :

- æ »»» ae
- ç »»» c
- η »»» n
- $\frac{2}{5}$  »»» 2/5
- Ø »»» O
- ....

### **Élimination des textes qui contiennent des mots étrangers :**

Comme nous l'avons déjà précisé au début de ce mémoire, nous considérons ce travail dans le cadre de l'anglais. Nous avons remarqué l'existence des mots étrangers à cette langue dans les textes du dataset, aussi nous avons décidé de supprimer tous les textes qui contiennent de tels mots au lieu de supprimer juste les mots pour éviter de créer un décalage sur les positions de début et fin des réponses.

The Yuan dynasty (Chinese: 元朝; pinyin: Yuán Cháo), officially the Great Yuan (Chinese: 大元; pinyin: Dà Yuán; Mongolian: Yehe Yuan Ulus[a]), was the empire or ruling dynasty of China established by Kublai Khan, leader of the Mongolian Borjigin clan. Although the Mongols had ruled territories including today's North

FIGURE 2.3 – Exemple d'un texte qui contient des mots étrangers

### **Tokenisation :**

L'étape de tokenisation consiste à transformer un texte en une liste de mots (tokens). Pour cela nous avons utilisé la fonction word\_tokenize de l'outil NLTK (Natural Langage ToolKit).

Exemple :

#### **Texte Normal**

Computational complexity theory is a branch of the theory of computation in theoretical computer science

#### **Texte tokenisé**

[‘Computational’, ‘complexity’, ‘theory’, ‘is’, ‘a’, ‘branch’, ‘of’, ‘the’, ‘theory’, ‘of’, ‘computation’, ‘in’, ‘theoretical’, ‘computer’, ‘science’]

### **Création d'un vocabulaire et Vectorisation :**

Le modèle qui va être entrainé a besoin d'un vocabulaire de mots pour les reconnaître dans la partie word embeddings. Ce vocabulaire est sous forme d'un grand dictionnaire d'index dans lequel chaque mot est associé à un index.

Exemple :

```
{ 'the' : 1, 'students' : 2, 'school' : 3, 'work' : 4, 'father' : 5, 'is' : 6, 'city' : 7, 'computer' : 8, 'with' : 9, 'a' : 10 }
```

Les textes et questions sont alors représentés en index pour les préparer au word embeddings.

Exemple :

**Texte :** The students work with a computer

**Texte indexé :** [1,2,4,9,10,8]

L'étape suivante consiste à définir une taille maximale des vecteurs car chaque texte peut avoir différentes tailles. Pour résoudre ce problème nous devons mettre tous les vecteurs à la même taille définie au préalable. Nous définissons la taille d'un vecteur d'un texte à 450 et d'une question à 150.

Exemple :

**Texte indexé :** [ 1 , 2 , 4 , 9 , 10 , 8 ]

**Texte avec la taille maximale :** [ 1 , 2 , 4 , 9 , 10 , 8 , 0 , 0 , 0 , 0 , 0 , 0 ... (x450) ]

La dernière étape est d'encoder les réponses en One-Hot-Encoding, ce qui consiste à attribuer la valeur 1 aux positions des mots début et fin de la réponse, et 0 à toutes les autres.

Exemple :

**Texte :** The students work with a computer

**Question :** With what the students work?

**Réponse :** A computer

**Début de la réponse (One-hot) :** [ 0 , 0 , 0 , 0 , 1 , 0 ]

**Fin de la réponse (One-hot) :** [ 0 , 0 , 0 , 0 , 0 , 1 ]

Maintenant que les données du dataset ont bien été filtrées, vectorisées, encodées, elles sont prêtes à être utilisé par modèle pour l'entraînement.

## 2.2 Modèles proposés

Dans cette section, nous présentons les différents modèles que nous proposons afin d'atteindre notre objectif.

### 2.2.1 Modèle de distance

Pour essayer de trouver une réponse à une question dans un texte, une réflexion simple serait de trouver les mots de la question dans le texte ; mais cela n'est pas suffisant pour trouver les mots contenus dans la réponse, ni même leurs positions.

Pour améliorer cette réflexion, il serait plus judicieux d'essayer de trouver la phrase contenue dans le texte la plus proche syntaxiquement et sémantiquement de la question. Cette phrase pourrait être celle qui contient potentiellement la réponse. Le fonctionnement de ce modèle est comme suit :

- Décomposition du texte en phrases.
- Word Embedding et Sentence Embedding.
- Fonction de distance.
- Sélection de la phrase du texte qui est la plus proche de la question.

#### Décomposition du texte en phrases :

Le principe est très simple : on divise le texte en plusieurs phrases, cela peut se faire avec plusieurs outils de traitement de langage naturel comme NLTK ou TEXTBLOB .

Exemple :

**Texte :** """ Architecturally, the school has a Catholic character. The Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary."""

**Texte décomposé :** "Architecturally, the school has a Catholic character."

"The Main Building's gold dome is a golden statue of the Virgin Mary."

"Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes" "

"Next to the Main Building is the Basilica of the Sacred Heart."

....

### Word embedding et Sentence embedding :

Afin de calculer les distances entre les phrases du texte et la question, nous commençons par représenter la question et chaque phrase par un vecteur. Il existe un modèle ([InferSent](#)) crée par l'entreprise FACEBOOK pour faire ce qu'on appelle "Sentence Embedding" et qui permet de représenter une phrase par un vecteur en prenant en compte le contexte syntaxique et sémantique de celle-ci, en combinant les représentations de word embedding des mots.

Ce modèle entrainé pour la tâche NLI (Natural Language Inference) peut être utilisé pour d'autres tâches, notamment la représentation de phrases (Sentence Embedding).

Avec ce modèle on peut représenter les phrases du texte ainsi que la question pour appliquer des fonctions de distances sur leurs valeurs.

### Fonction de distance

Dans cette partie, nous calculons une distance pour chaque vecteur de phrase avec le vecteur de la question, la distance la plus courte pourra déterminer la phrase la plus proche de la question et qui contiendra potentiellement la réponse.

Exemple de fonctions de distances :

- Distance euclidienne :  $d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$ , plus la distance est petite plus les deux vecteurs sont proches.

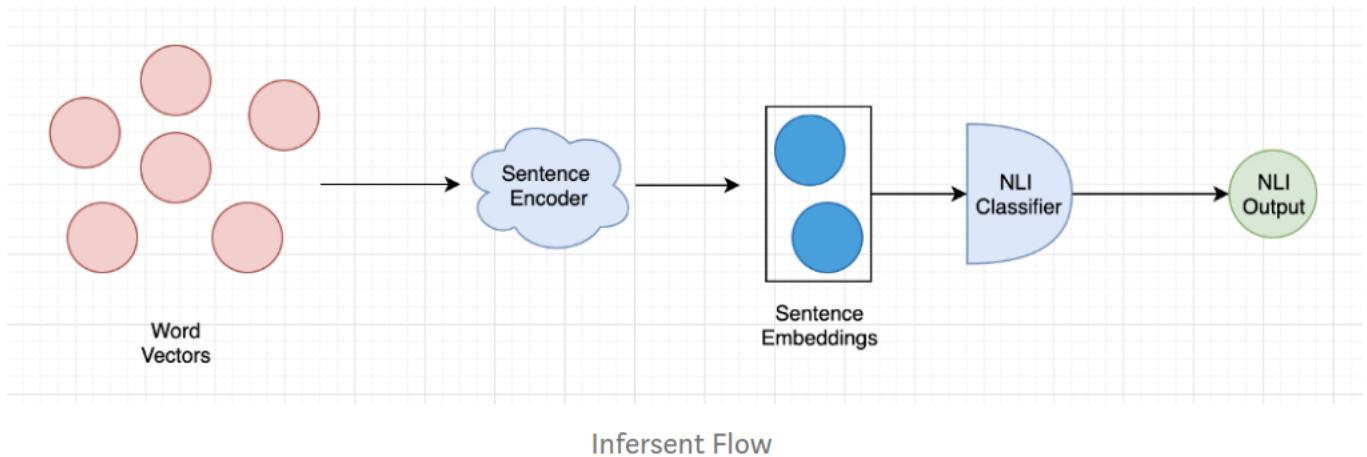


FIGURE 2.4 – Fonctionnement du modèle InferSent de FACEBOOK

**Source:** [www.analyticsvidhya.com](http://www.analyticsvidhya.com)

— Similarité de cosinus :

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{t}\mathbf{e}}{\|\mathbf{t}\| \|\mathbf{e}\|} = \frac{\sum_{i=1}^n \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^n (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^n (\mathbf{e}_i)^2}} \quad (2.1)$$

plus la valeur est grande plus les deux vecteurs sont proches.

#### Selection de la plus proche phrase du texte de la question :

Une fois les distances entre chaque phrase du texte et la question calculées, nous pourrons choisir celle qui est la plus proche de la question et qui pourrait contenir potentiellement la réponse.

Exemple tiré du dataset SQuAD :

**Texte :** """ Architecturally, the school has a Catholic character. The Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. **It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858.** At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary."""

**Question :** "To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?"

**Réponse exacte : Saint Bernadette Soubirous**

**Phrases candidates :**

Num de phrase	distance euclidienne	phrase
1	0.66	Architecturally, the school .... character.
2	0.54	The Main Building's .... Mary.
3	0.69	Immediately in front of .... "Venite Ad Me Omnes".
4	0.59	Next to the Main .... Sacred Heart.
5	0.65	Immediately behind the basilica .... and reflection.
<b>6</b>	<b>0.33</b>	<b>It is a replica of the grotto .... in 1858.</b>
7	0.69	At the end of the main .... statue of Mary.

La phrase 06 étant la plus proche phrase et contient effectivement la réponse.

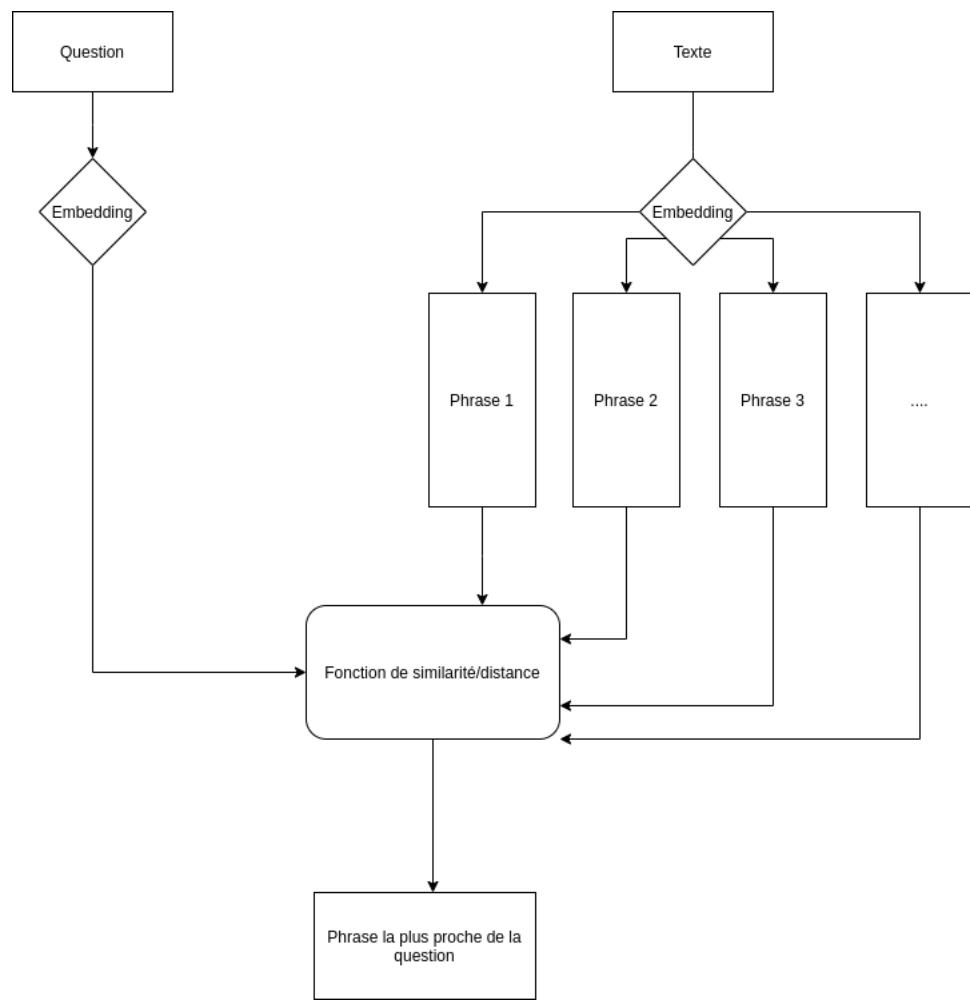


FIGURE 2.5 – Architecture générale du modèle de Question Answering basé sur la distance

## 2.2.2 Modèle basé LSTM (avec et sans mécanisme d'attention) :

Deux modèles sont proposés ici et qui ont globalement la même architecture mise à part l'ajout de mécanismes d'attention dans le deuxième modèle.

On peut décomposer les deux modèles en plusieurs parties comme suit :

- Word embeddings.
- Extraction des vecteurs de contextes.
- Matrice de similarité et mécanisme d'attention. (modèle avec attention)
- Concaténation globale et couches de sortie.

### i Partie 1 : Word embeddings

Comme dans le modèle précédent, on utilise le modèle pré-entraîné Glove pour produire une représentation en vecteur numérique des mots du texte et de la question en prenant en compte la similarité syntaxique et sémantique des mots.

### ii Partie 2 : Extraction des vecteurs de contextes

Les vecteurs de représentations des mots du texte et de la question passent chacun dans des réseaux de neurones récurrents de type LSTM pour générer des vecteurs de contextes. Ces vecteurs sont la représentation du contexte de chaque mot dans la suite de mots qui le précédent et le suivent.

- $C_T$  : vecteur de contexte du texte,  $T \in \mathbb{N}$  représente le nombre de mots dans le texte .
- $C_Q$  : vecteur de contexte de la question,  $Q \in \mathbb{N}$  représente le nombre de mots dans la question.

### iii Partie 3 : Matrice de similarité et mécanisme d'attention

Le but de cette approche est de fusionner les deux vecteurs des contextes du texte et de la question pour créer plusieurs représentations matricielles du texte qui contiennent également des informations importantes de la question.

Le mécanisme d'attention permet de se focaliser que sur quelques mots importants pendant l'entraînement, ce qui permet d'avoir plus de précision et plus de rapidité. Il en existe plusieurs versions différentes de mécanisme d'attention. Pour ce modèle nous appliquons le mécanisme d'attention introduit par des chercheurs de l'université de Washington dans leurs recherches en question answering avec leur Modèle BIDAF (Bidirectional Attention Flow) [16].

L'approche consiste à construire une matrice de similarité et produire deux matrices d'attention. La matrice de similarité  $M_S$  est une matrice de taille  $T * Q$  ( $T$  étant le nombre de mots du texte et  $Q$  le nombre de mots de la question), la valeur d'un élément de la ligne  $t$  et colonne  $q$  représente la similarité entre le  $t^{\text{ème}}$  mot du texte et le  $q^{\text{ème}}$  mot de la question.

La formule pour calculer cette matrice est la suivante :

$$M(a, b) = W^T * [a; b; a \circ b]$$

- $a$  et  $b$  sont deux vecteurs en entrée.
- $W^T$  est un vecteur de poids initialisé aléatoirement.
- $[;]$  est la concaténation de vecteurs.
- $\circ$  est le produit scalaire de deux vecteurs.

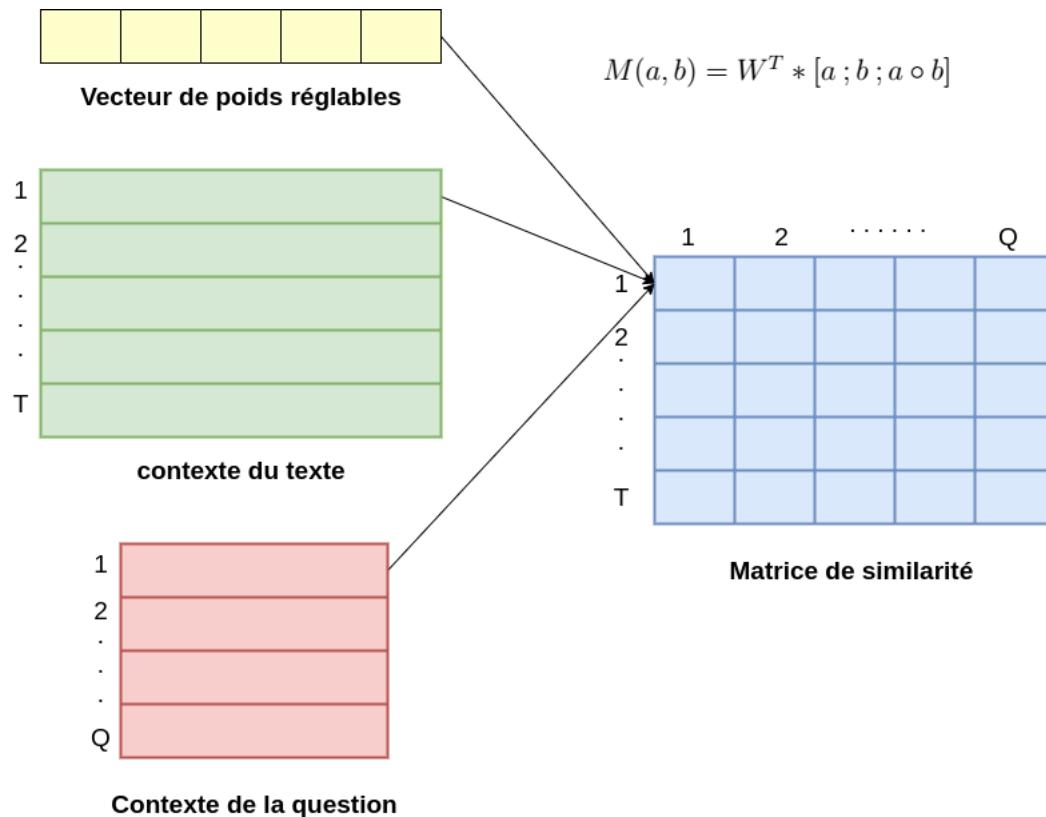


FIGURE 2.6 – Schéma représentant la construction de la matrice de similarité.

Prenons un exemple d'une matrice de similarité :

- Texte : " Singapore is a big country located in north africa "
- Question : " where is singapore situated "

La matrice de similarité résultante est la suivante :

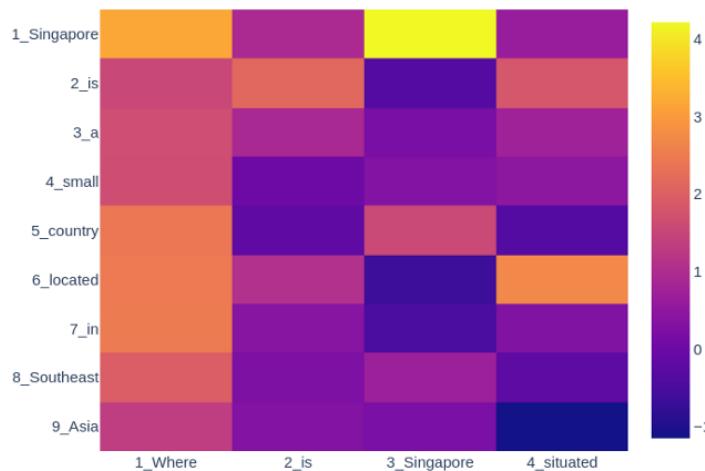


FIGURE 2.7 – Exemple de matrice de similarité

Source: [towardsdatascience.com](https://towardsdatascience.com/word2vec-similarity-matrix-101-10f3a2a2a2d)

- Comme nous pouvons le voir les deux mots "Singapore" ont logiquement des vecteurs de représentation similaires.
- Nous remarquons également que la paire du mot "is" n'est pas aussi élevée du fait que les contextes des deux mots ne sont pas les mêmes.
- Nous pouvons voir que les mots "situated" et "located" sont très similaires.

Donc cette matrice de similarité permet de prendre en compte la similarité syntaxique, sémantique, et contextuelle qui sont très importantes pour comprendre la relation entre le texte et la question. Nous pouvons maintenant construire les deux matrices d'attention texte-question et question-texte.

- Matrice d'attention Question-Texte : elle permet de déterminer les mots de la questions les plus importants par rapport aux mots du texte.

Sa formule de calcul :

- Pour chaque ligne de la matrice de similarité, calculer la somme pondérée avec le contexte de la question puis dupliquer le vecteur résultat pour avoir une matrice de même taille que la matrice d'attention.

- Matrice d'attention Texte-Question : elle permet de déterminer les mots les plus importants du texte.

Sa formule de calcul :

- Extraire un vecteur qui contient le maximum de chaque ligne de la matrice de similarité, puis calculer une somme pondérée avec le contexte du texte.

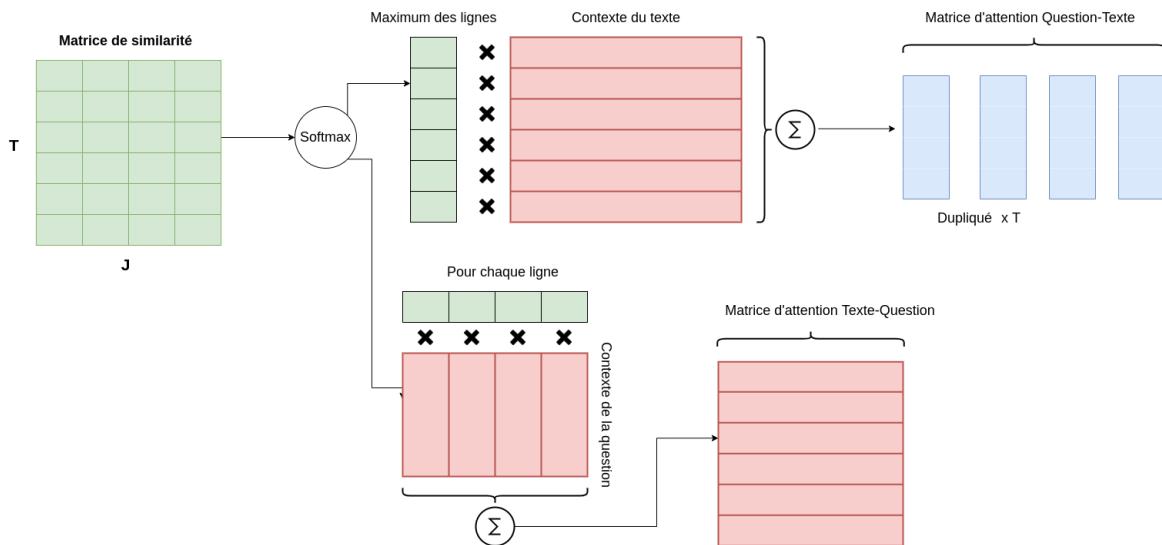


FIGURE 2.8 – Schéma de constructions des deux matrices d'attention

#### iv Partie 4 : Concaténation globale et Couches de sortie

**Pour le modèle sans mécanisme d'attention :** Après l'extraction des deux vecteurs des contextes du texte et de la question, nous les concaténonons directement et les insérons dans un décodeur LSTM pour prédire directement les mots de la réponse.

**Pour le modèle avec mécanisme d'attention :** Après la construction des deux matrices, nous les concaténons avec le contexte du texte pour ensuite envoyer le résultat vers une couche de réseaux de neurones LSTM pour en extraire un vecteur de contexte global qui prend en considération toutes les nouvelles données importantes grâce au mécanisme d'attention.

La toute dernière étape est d'insérer ce vecteur résultant dans des couches denses avec une fonction activation softmax, chacune a pour but de prédire les positions des début et fin de la réponse.

#### v L'entraînement

Nous avions 4 principaux paramètres à varier afin de trouver une combinaison satisfaisante qui sont :

- Nombre de neurones (entre 32 et 256).
- Nombre d'itérations (entre 5 et 300).
- Différentes fonctions d'optimisation de l'entraînement de tensorflow (ADAM, SGD, RMSprop ...).
- Taux d'apprentissage (Learning rate) (entre 0,0 et 1,0) : il s'agit d'un paramètre dans une fonction d'activation qui permet entre autre de réguler la vitesse d'apprentissage d'un modèle, plus le taux est petit plus le modèle va prendre son temps à mieux apprendre et réciproquement.

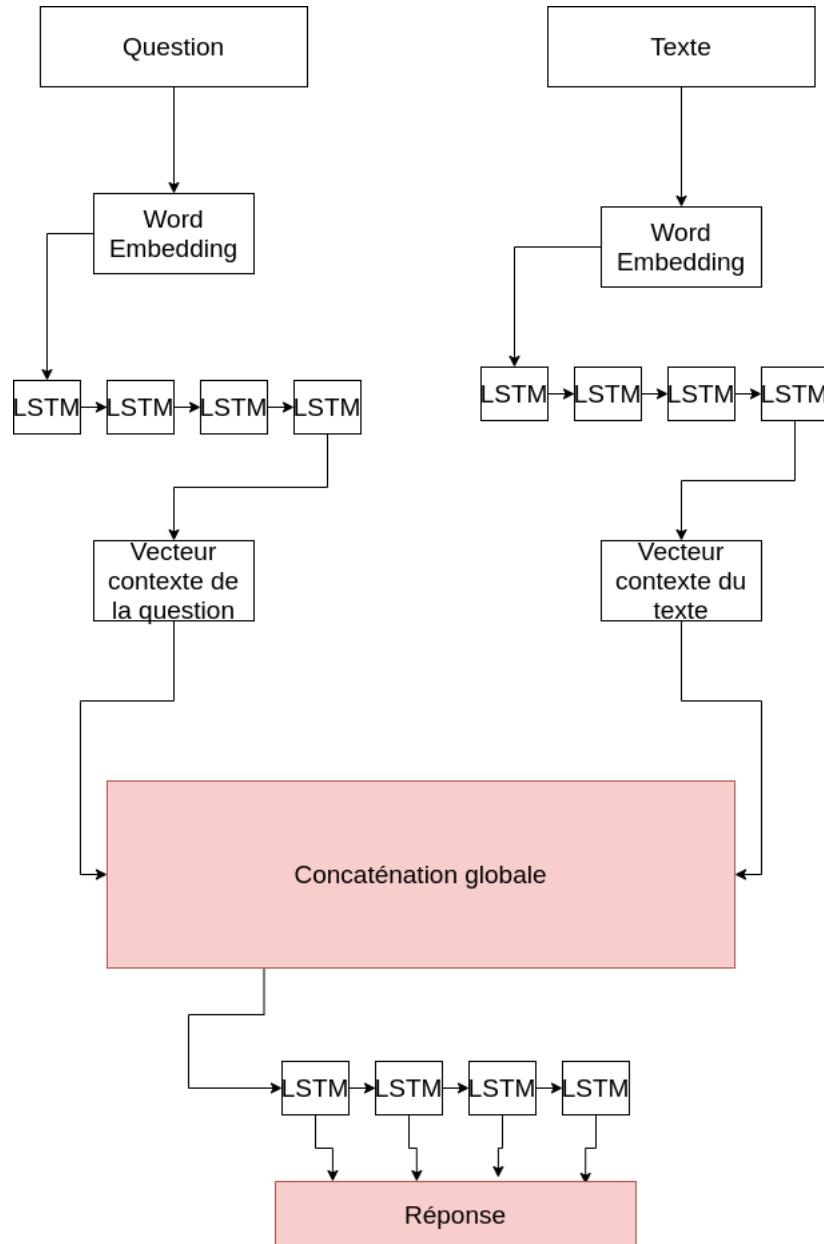


FIGURE 2.9 – Architecture du modèle LSTM sans attention

Nous avons mis le code principal de l’entraînement dans des boucles imbriquées pour varier les différents paramètres, et nous avons activé une fonction qui permet d’arrêter l’entraînement en cours si les résultats du modèle ne s’améliorent pas (Early Stopping de Keras) et ainsi passer à une autre combinaison de paramètres sans perdre de temps. Au final, nous ne gardons que l’architecture qui a eu les meilleurs résultats parmi toutes les autres. Les résultats de l’entraînement du dernier modèle retenu sont alors gardés dans un fichier Excel afin de créer les graphes correspondants.

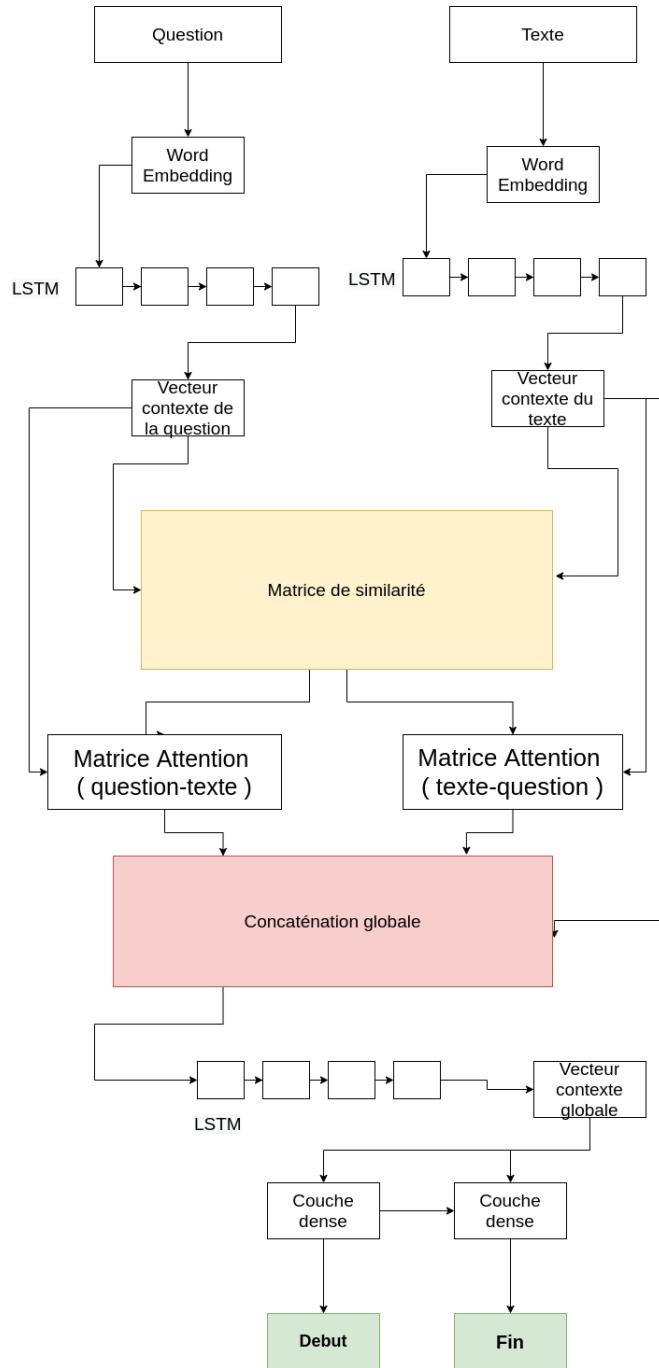


FIGURE 2.10 – Architecture du modèle LSTM avec attention

### 2.2.3 Modèle d'apprentissage par transfert (transfer learning)

Le modèle pré-entraîné choisi parmi d'autres se nomme BERT (Bidirectional Encoder Representation from Transformers). C'est un modèle de représentation de langage qui consiste, comme son nom l'indique, à comprendre et à représenter un langage à partir d'un texte brut. Ce qui permet entre autre à extraire un contexte et à encoder un texte de manière optimisée.

Ce modèle a été entraîné par l'entreprise GOOGLE sur de très grands corpus de MLM (Masked Language Model) qui permet de prédire des mots manquants dans une phrase et NSP (Next Sentence Prediction) qui permet prédire si une phrase vient après une autre, deux principales tâches pour comprendre un langage. BERT est composé de plusieurs couches d'encodeurs, d'une architecture appelée 'Transformer' et qui ont pour but de modéliser et de représenter un texte et d'en extraire son contexte.

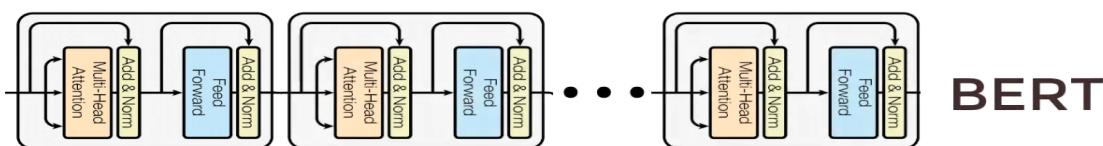


FIGURE 2.11 – Architecture de BERT (Multi-couches d'encodeurs de Transformer)

Il a en entrée un élément qui est la concaténation de deux arguments de forme textuelle brute séparés par un séparateur pour être encodés en trois étapes. Les données peuvent ainsi passer dans plusieurs couches d'encodeurs pendant l'entraînement pour avoir en sortie du modèle la prédiction des deux tâches MLM (les mêmes arguments avec prédiction de mots manquants) et NSP (Une variable booléenne  $C$  qui vérifie si l'argument 2 est bien après l'argument 1)

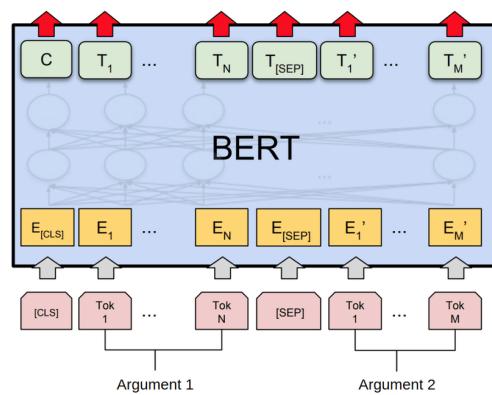


FIGURE 2.12 – Entrées et sorties de BERT

L'un des avantages de BERT est la partie word embeddings pour encoder les mots, qui se compose comme suit :

- Bert Tokenizer : un outil intégré au modèle pour tokeniser un texte et extraire les mots et ajouter le séparateur entre les deux deux textes.
- Token embeddings : qui consiste à représenter les mots en vecteurs.
- Segment embeddings : qui consiste à donner un code pour les arguments en entrée pour chacun de leurs mots.
- Position embeddings : qui représente la position de chaque mot dans l'argument en entrée.
- Comme expliqué précédemment sur le mécanisme d'attention dans l'architecture Transformer, comme l'architecture de BERT est composée de plusieurs couches d'encodeurs d'un transformer, alors il dispose de plusieurs couches d'attention, ce que les chercheurs appellent " Self Attention ". En résumé, chaque mot du texte et de la question a un score d'attention avec eux-mêmes et avec le reste des mots.

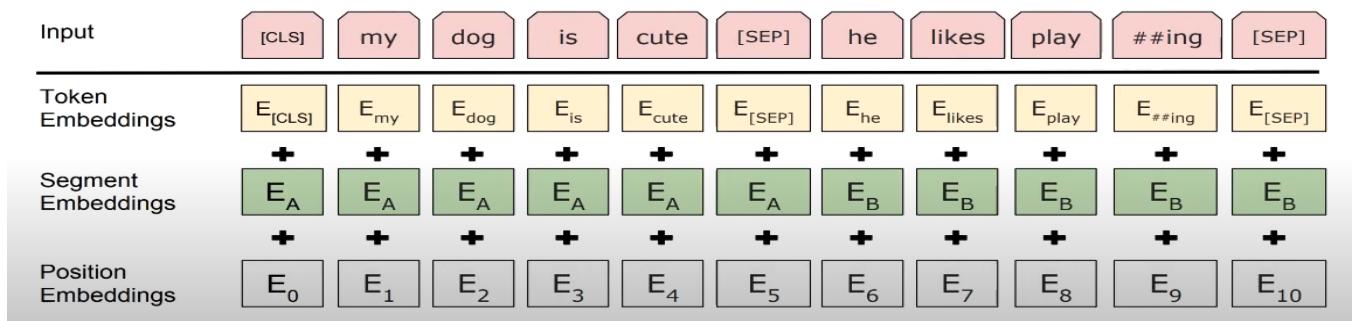


FIGURE 2.13 – Schéma de la partie embeddings de BERT

## Fine Tuning de BERT

Comme expliqué précédemment, nous allons utiliser la méthode *Fine-Tuning* sur le modèle BERT, qui consiste à laisser toutes les couches du modèle qui peuvent être mises à jour et de remplacer la dernière couche de prédiction par deux nouvelles couches denses avec activation softmax pour calculer la distribution de probabilités qui correspondent aux positions début et fin de la réponse.

Nous avons appliqué le *Fine-Tuning* de BERT de la manière suivante :

- Tokeniser le corpus avec le module de tokenisation de BERT, ce qui va retourner :
  1. le texte et la question tokénisés de cette forme : [Tmot1, Tmot2, Tmot3, ..., 'SEP', Qmot1, Qmot2, Qmot3 ...]
  2. Un vecteur d'index pour chaque mot et un vecteur d'appartenance de chaque mot soit pour le texte, soit pour la question.
  3. Des vecteurs d'attention initiaux au début et qui vont s'entrainer durant l'entraînement.
- Insérer les résultats de la précédente étape dans l'encodeur du modèle pré-entraîné BERT pour extraire des vecteurs d'états cachés de la dernière couche de l'encodeur.
- Les résultats vont alors passer par deux couches denses (une pour la prédiction du début de la réponse et une pour la prédiction de la fin de la réponse) avec une fonction d'activation softmax pour calculer les probabilités qu'un mot soit le début ou la fin.

Nous n'avons varié aucun paramètre, le modèle apprenait très vite en un nombre très petit d'itérations et ne semblait pas s'améliorer si on rajoutait des itérations. Et comme la plus grande partie du modèle était basée sur un modèle BERT pré-entraîné, nous ne pouvions pas varier les paramètres internes de celui-ci mise à part les poids du modèle durant l'entraînement.

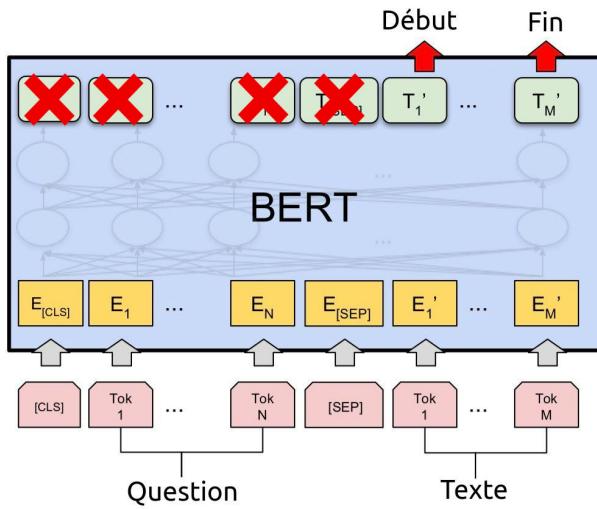


FIGURE 2.14 – Nouveau modèle adapté au Question Answering en utilisant BERT

Le modèle a en entrée la question et le texte concaténés qui passent par plusieurs couches d'encodeurs de BERT pour créer une représentation en vecteur, ensuite ils passent par deux couches denses afin de calculer la probabilité de chaque mot. Les mots ayant les plus grandes probabilités dans les couches START et END seront respectivement les positions du début et la fin de la réponse dans le texte.

## 2.3 Résultats d'entraînement

### i Modele LSTM sans attention

#### Détails du modèle et de l'entraînement

- Nombre de neurones par couche : 64
- Nombre d'itérations : 128
- Temps d'entraînement par itération : ~ 40 min

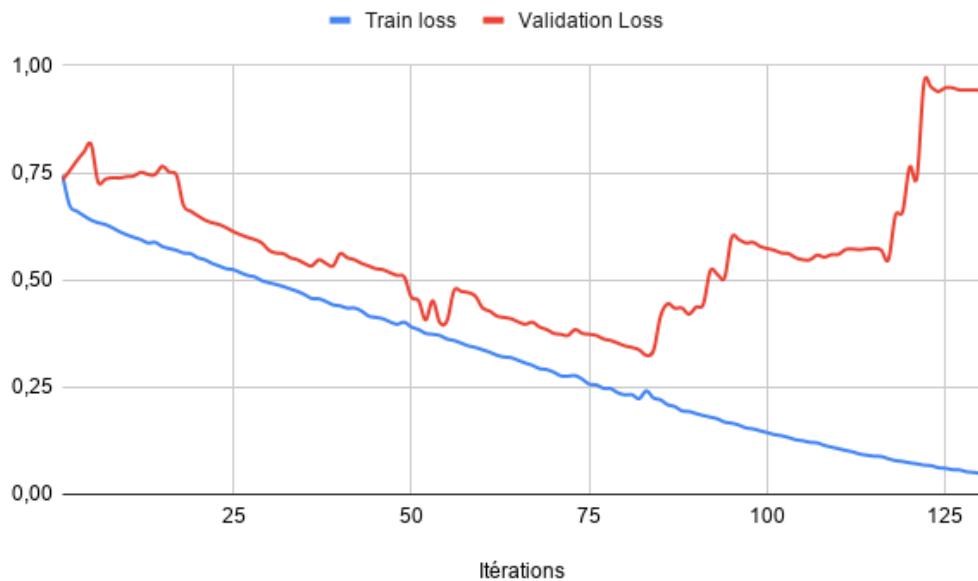


FIGURE 2.15 – Courbes d’erreur pendant l’entraînement et le test

Nous remarquons clairement que pendant la validation l’erreur augmente et donc que l’entraînement s’est mal déroulé. Nous avons effectué cet entraînement plusieurs fois mais nous avons toujours obtenu approximativement la même courbe pour plusieurs valeurs des paramètres du modèle. Nous avons malgré cela fait le choix de garder ce modèle pour la phase de test afin d’essayer d’analyser et d’expliquer les causes de ce mauvais entraînement. Notre intuition derrière ce choix est que cela pouvait nous permettre de chercher un moyen pour améliorer son architecture et arriver à avoir des débuts de résultats intéressants.

## ii Modele LSTM avec attention :

### Détails du modèle et de l’entraînement :

- Nombre de neurones par couche : 128
- Nombre d’itérations : 16
- Temps d’entraînement par itération : ~ 18 min

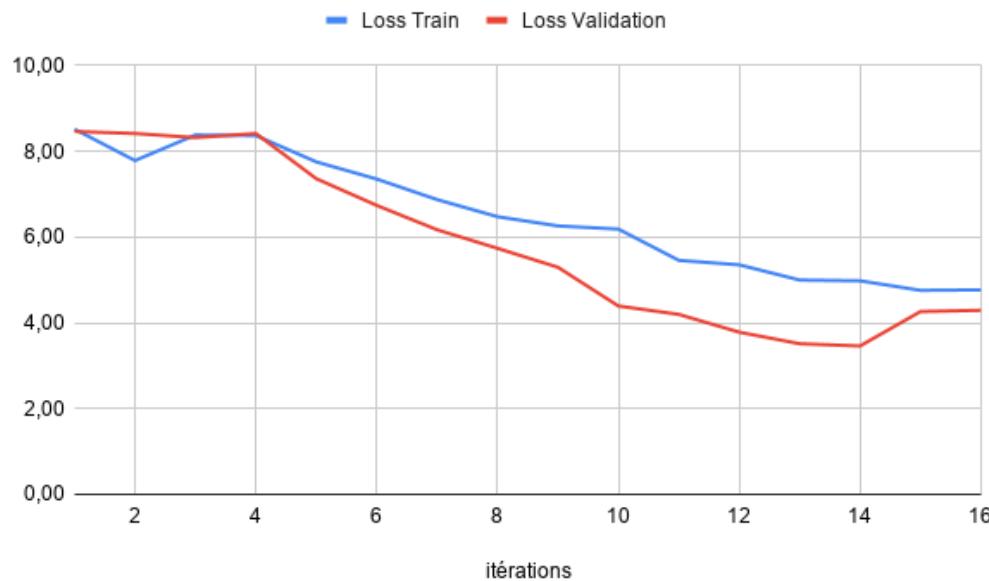


FIGURE 2.16 – Courbes d'erreur pendant l'entraînement et le test

Nous remarquons ici que les courbes d'erreur sont nettement meilleures que dans le précédent modèle : les valeurs de l'entraînement et de la validation se rapprochent mais semblent ne plus diminuer vers la fin. Nous avons ainsi appliqué une fonction appelée "" Early stopping "" qui permet d'arrêter l'entraînement avant sa fin si l'outil estime que les valeurs commencent à ne plus être satisfaisantes. L'entraînement s'arrête à la seizeième itération en estimant que le modèle ne pourrait plus apprendre davantage.

### iii Modèle par apprentissage par transfert

#### Détails du modèle et de l'entraînement :

- Nombre de couches de BERT : 12
- Nombre de neurones par couche : 768
- Taille maximale d'entrée : 512 tokens (mots)
- Nombre d'itérations : 12
- Temps d'entraînement par itérations : ~ 5 min

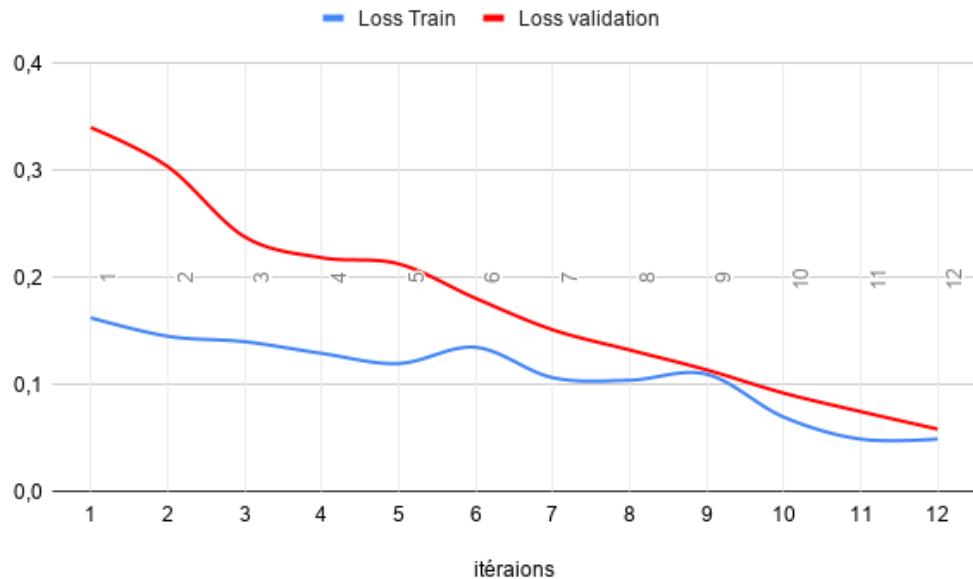


FIGURE 2.17 – Courbes d’erreur pendant l’entraînement et le Test

Nous remarquons que ces courbes d’erreur sont de loin les meilleurs par rapport aux modèles précédents. L’erreur de l’entraînement et de la validation diminuent et se rapprochent jusqu’à ce qu’elles stagnent et permettre de décider de s’arrêter là.

Nous avons présenté les différentes solutions que nous proposons pour le problème du Question Answering. Le prochain chapitre est dédié aux tests des différents modèles ainsi que leur mise en œuvre sous forme d’une application web.

# CHAPITRE 3

## IMPLÉMENTATION ET TESTS

Dans ce chapitre nous présentons l'environnement de développement et d'expérimentation de notre application ainsi que les différentes technologies mises en œuvres afin d'implémenter les modèles proposés. Les résultats sont montrés par les différents tableaux et par les graphes. Ils seront par la suite discutés.

### 3.1 Environnement de développement et d'expérimentation

#### 3.1.1 Plateformes, outils et langages de programmation

Nous avons utilisé le langage de programmation python pour la construction, l'entraînement et les tests des modèles, ainsi que le langage Javascript<sup>1</sup> avec pour interface de l'application et le déploiement des modèles.

Pour les traitements sur le corpus comme le nettoyage et la tokénisation nous avons utilisé l'outil NLTK (Natural Langage Tool Kit) qui possède des fonctions très utiles pour le traitement de textes et qui reste l'un des outils les plus connus dans le domaine du TALN.

Pour la partie WORD EMBEDDING nous avons utilisé le modèle pré-entraîné GLOVE pour permettre la vectorisation des mots en prenant en compte la sémantique, ainsi que pour le modèle de distances nous avons utilisé l'outil InferSent de FACEBOOK pour vectoriser des phrases en prenant en compte leurs sémantique toujours.

Pour tout ce qui est calculs mathématiques et manipulations de vecteurs, nous avons utilisé les librairies python Numpy, Scipy ainsi que matplotlib pour afficher des schémas graphiques.

1. la bibliothèque Eel : <https://martin-thoma.com/eel/>

Pour la création des modèles et leurs manipulations, nous avons utilisé les frameworks TensorFlow et Keras. Ce sont des outils puissants, pratiques et efficaces pour construire des modèles d'apprentissage profond comme les réseaux de neurones.



FIGURE 3.1 – Logos de quelques outils utilisés

L'entraînement des modèles a été fait dans la plateforme Google Cloud Plateform et Google Co-lab qui mettent à disposition des machines virtuelles relativement plus puissantes que de simples machines afin d'exécuter l'entraînement de nos modèles.

La puissance de la machine est un paramètre très important pour le temps d'entraînement, et les caractéristiques de la machine mises à notre disposition par GOOGLE sont les suivantes :

TABLE 3.1 – Les caractéristiques de la machine Google Colab que nous avons utilisé pour entraîner nos modèles.

Paramètres	Google Colab
Modèle carte graphique	GPU NVIDIA K80 / T4
Puissance GPU	0.82GHz / 1.59GHz
Mémoire vive	12 Go
Date de sortie	2014-2016
Nombre de coeurs	2
Espace disque	350 Go

## 3.2 Tests et résultats

### 3.2.1 Corpus de tests

Pour tester la performance de nos modèles, nous avons exploité le corpus SQuAD qui contient déjà une partie dédiée au test qui représente environ 15% du corpus global. Soit plus précisément 87599 paires (texte, question) pour l'entraînement et 14725 pour le test.

Pour ce qui est de la distribution des questions par type dans le corpus de test, nous pouvons voir qu'elle reste assez équilibrée pour effectuer les tests par rapport à la distribution globale de tout le corpus.

TABLE 3.2 – Répartition des questions dans le corpus de tests

Type de question	Nombres de pairs (texte,question)	Pourcentage
What	3681	25%
Who	3240	22%
Why	178	8%
How	3828	26%
Where	1031	7%
When	1767	12%

### 3.2.2 Mesures de performance

Pour tester la performance et l'aptitude des modèles à répondre aux questions, nous utiliserons les deux mesures suivantes :

- F-Mesure (F1-score) :

$$F1-score = \frac{2 * (precision * rappel)}{(precision + rappel)} \quad (3.1)$$

$$Precision = \frac{\text{Nombre de mots correctement prédits}}{\text{Nombre totale de mots prédits}}$$

$$Rappel = \frac{\text{Nombre de mots correctement prédits}}{\text{Nombre de mots sur la vraie réponse}}$$

- Exact match (EM) :

$$EM = \frac{\text{Nombre de réponses exactes correctes}}{\text{Nombre total de réponses}} \quad (3.2)$$

### 3.2.3 Résultats des tests

Les tests ont été appliqués sur le corpus de test en calculant pour chaque modèle les mesures de performances présentées précédemment.

TABLE 3.3 – Tableau des résultats

Modèle	EM	F1-score	Phrases proches (modele distance)
Distance euclidienne	0	9.36%	48,87%
Similarité Cosinus	0	11,7%	59,71%
LSTM sans Attention	0	0,45%	
LSTM avec Attention	58,53%	70,28%	
Transfert learning de BERT	75%	77%	
D'autres modèles connus			
SG-Net [17]	87.23%	90.07%	
Retro-Reader on ALBERT [18]	88.10%	91.41%	
ALBERT [19]	89.73%	92.21%	
Performance humaine (Selon l'équipe de SQuAD)	86.831	89.452%	

Pour le modèle basé sur le calcul de la distance nous avons décidé de mesurer si ce modèle arrive à trouver les phrases les plus similaires à celles qui contiennent la réponse. Par ce que en principe il cherche les phrases les plus similaires à la question.

## Discussion

- Les deux modèles de distances et le modèle LSTM sans attention n'ont pas de bons résultats par rapport au modèle LSTM avec attention et le modèle d'apprentissage par transfert de BERT, ce qui montre l'importance et l'efficacité du mécanisme d'attention.
- Le modèle basé sur le calcul des distances arrive quand même à trouver plus de la moitié des phrases qui contiennent la réponse. C'est un meilleur résultat par rapport au modèle LSTM sans attention qui ne trouve ni la réponse exacte ni les mots de la réponse. Il ne fait que mémoriser les données qu'il a déjà vu durant son entraînement, c'est pour cela qu'il trouve facilement les réponses pour les textes sur lesquels il s'est entraîné.

**Résultats par rapport aux questions** Les deux modèles les plus performants des quatre sont le modèle LSTM avec attention et le modèle d'apprentissage par transfert. Cela montre l'efficacité et l'importance d'un mécanisme d'attention qui existe dans les deux. Pour les comparer avec un peu plus de précision, nous avons analysé leurs performances (Exact match) sur plusieurs types de questions.

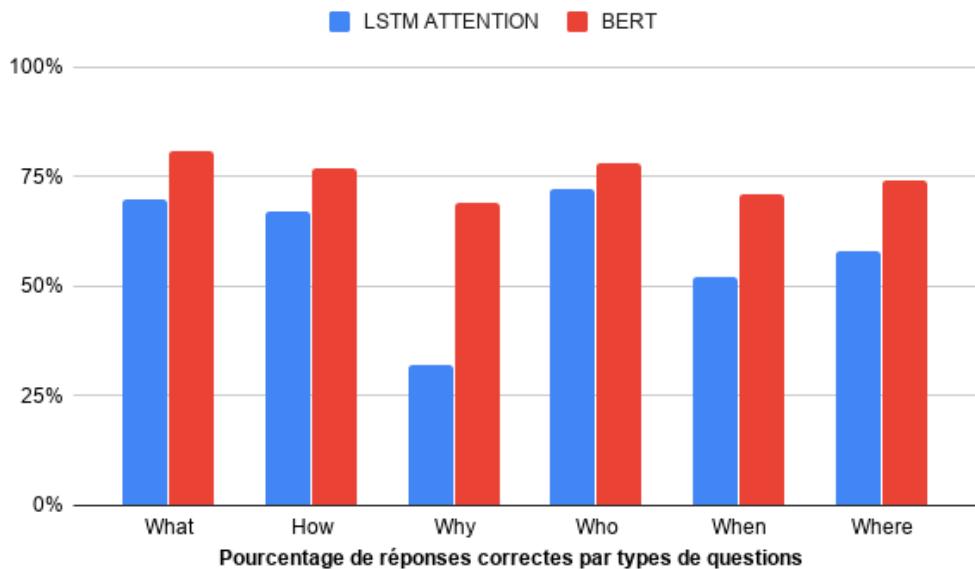


FIGURE 3.2 – Pourcentage de bonnes réponses par type de questions

- Nous remarquons que le modèle basé sur le Transfert Learning du BERT donne de meilleurs résultats dans toutes les questions.
- Les performances du modèle LSTM avec Attention se dégradent d'une manière remarquable avec les questions "Why".

Pour en savoir plus sur les deux modèles, prenons un simple exemple concret :

**Texte :** Algeria had an estimated population of over 44 million.

**Question :** how much people leave in algeria ?

**Réponse du modèle LSTM avec attention :** 44 million

**Réponse du modèle d'apprentissage par trasnfert :** 44 million

**Vraie réponse :** 44 million

Nous remarquons dans la matrice d'attention (Voir Figure 3.3) texte-question du modèle LSTM avec attention que les deux mots "how" et "much" sont très importants par rapport au texte pour prédire la réponse, de même pour les mots "estimated", "over", "44", "million". La puissance de l'attention est de se préoccuper de quelques mots précis et importants qui peuvent avoir une utilité pour la réponse.

La matrice d'auto attention (self attention) du modèle d'apprentissage par transfert (BERT) est différente (Voir Figure 3.4) mais donne aussi de l'importance aux mots "over" "44" "million" par rapport aux mots "estimated" et "populated".

**Disucssion** Le modèle BERT utilise des architectures de réseaux de neurones et un mécanisme d'attention différent par rapport à celui des réseaux LSTM. En plus, le mécanisme d'attention im-

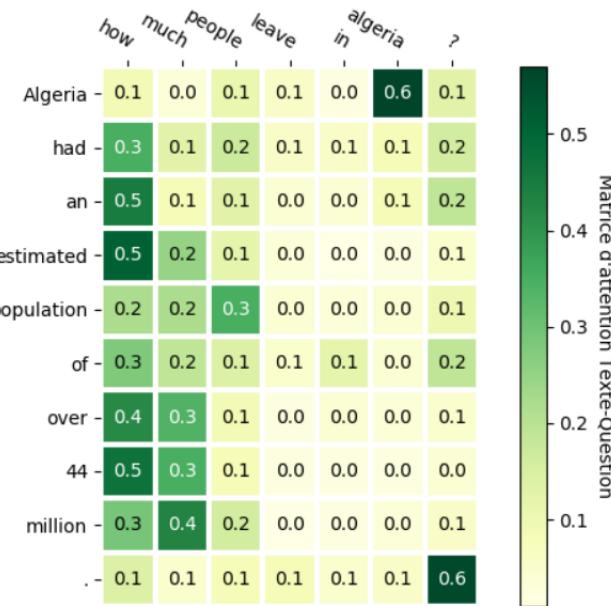


FIGURE 3.3 – Matrice d'attention du modèle LSTM avec attention

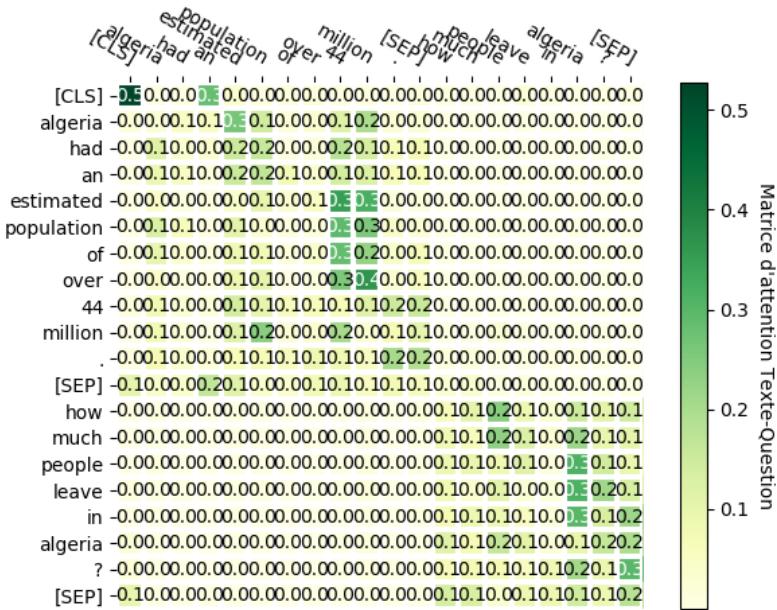


FIGURE 3.4 – Matrice d'auto attention (self attention) du modèle BERT

plémenté au niveau de BERT est plus robuste par rapport à celui du modèle LSTM ce qui a engendré cette différence dans les résultats.

**Temps d'exécution** Nous avons testé le temps moyen d'exécution que chaque modèle a besoin pour prédire une réponse.

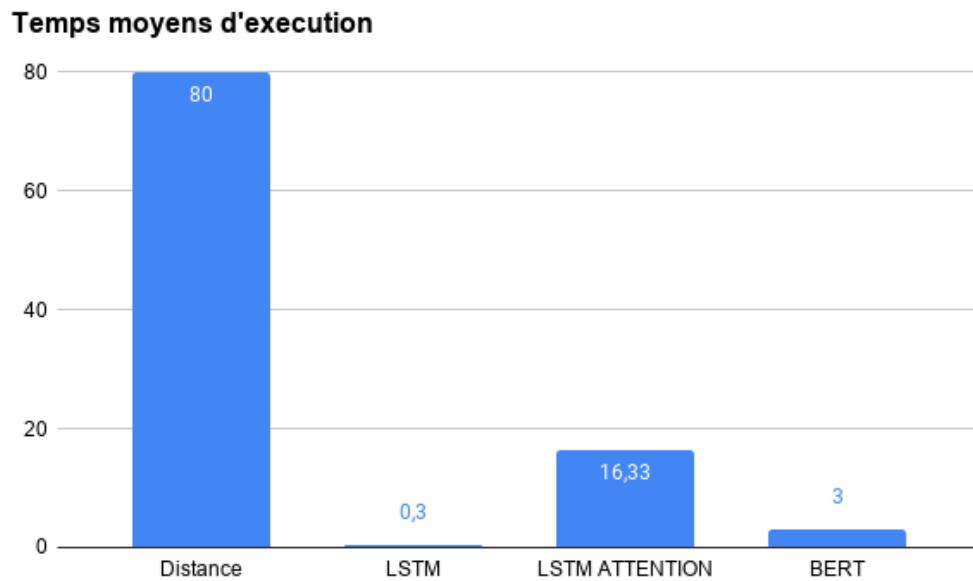


FIGURE 3.5 – Temps moyen d'exécution en secondes par modèle

Comme nous le voyons dans la figure 3.5, le modèle de distances est de loin le plus lent à répondre, et le modèle LSTM sans attention le plus rapide mais ne trouve pratiquement jamais de réponses. On voit alors que le modèle BERT est clairement le plus performant et le plus rapide à prédire une réponse satisfaisante.

### 3.3 Interface de l'application

L'interface de l'application se compose en 4 pages, une page pour chaque modèle, et toutes les pages ont en commun ces champs :

- Une barre de navigation pour changer de modèle.
- Champ de texte pour entrer le texte.
- Champ de texte pour la question.
- Un bouton pour lancer la prédiction de la réponse.
- Un bouton pour afficher des détails pertinents.
- Un espace pour afficher la réponse et quelques caractéristiques.

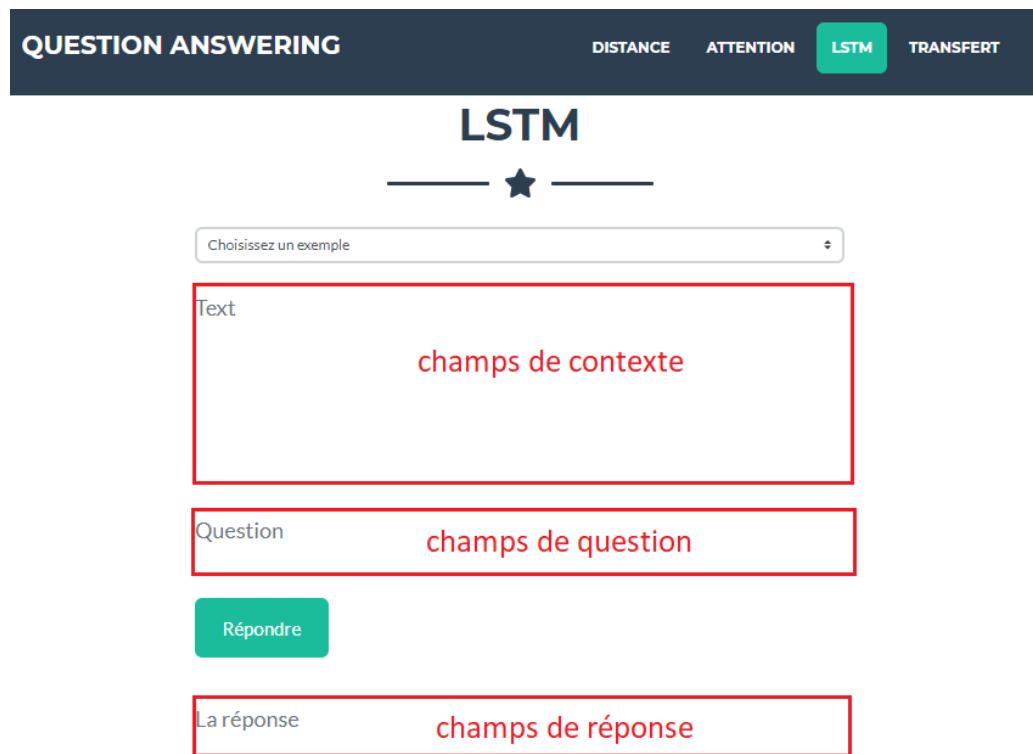


FIGURE 3.6 – Interface de la page d'accueil

En dessous du champs libre du texte, un sélecteur permet le choix parmi les textes déjà enregistrés du corpus ou bien sélectionner de nouveaux textes pour les tester directement.

exemple 1

Le texte

Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend Venite Ad Me Omnes. Next to the Main Building is the Basilica of the

La question

What is in front of the Notre Dame Main Building?

FIGURE 3.7 – Sélectionner un texte pour le test

En cliquant sur le bouton Répondre, la réponse s'affiche ainsi que le bouton afficher les détails apparaît.

Le texte

Philosophers in antiquity used the concept of force in the study of stationary and moving objects and simple machines, but thinkers such as Aristotle and Archimedes retained fundamental errors in understanding force. In part this was due to an incomplete understanding of the sometimes non-obvious force of friction, and

La question

Who developed the theory of relativity?

Répondre

Afficher les détails

La réponse

Einstein

FIGURE 3.8 – Test de réponse sur l'interface

**L'affichage de détails** Nous pouvons afficher quelques détails pertinents en appuyant sur le bouton "Afficher les détails".

**L'affichage du passage du texte contenant la réponse** La réponse est affichée en rouge dans le texte ainsi que le temps d'exécution.

Le temps d'execution est de 23876 ms

**La question:**

Who developed the theory of relativity?

**La réponse dans le contexte :**

philosophers in antiquity used the concept of force in the study of stationary and moving objects and simple machines, but thinkers such as aristotle and archimedes retained fundamental errors in understanding force. in part this was due to an incomplete understanding of the sometimes non-obvious force of friction, and a consequently inadequate view of the nature of natural motion. a fundamental error was the belief that a force is required to maintain motion, even at a constant velocity. most of the previous misunderstandings about motion and force were eventually corrected by galileo galilei and sir isaac newton. with his mathematical insight, sir isaac newton formulated laws of motion that were not improved-on for nearly three hundred years. by the early 20th century, Einstein developed a theory of relativity that correctly predicted the action of forces on objects with increasing momenta near the speed of light, and also provided insight into the forces produced by gravitation and inertia.

FIGURE 3.9 – Affichage de quelques détails d'une réponse

**L'affichage de la matrice des distances** Dans le cas du modèle LSTM avec attention, la matrice d'attention est affichée.

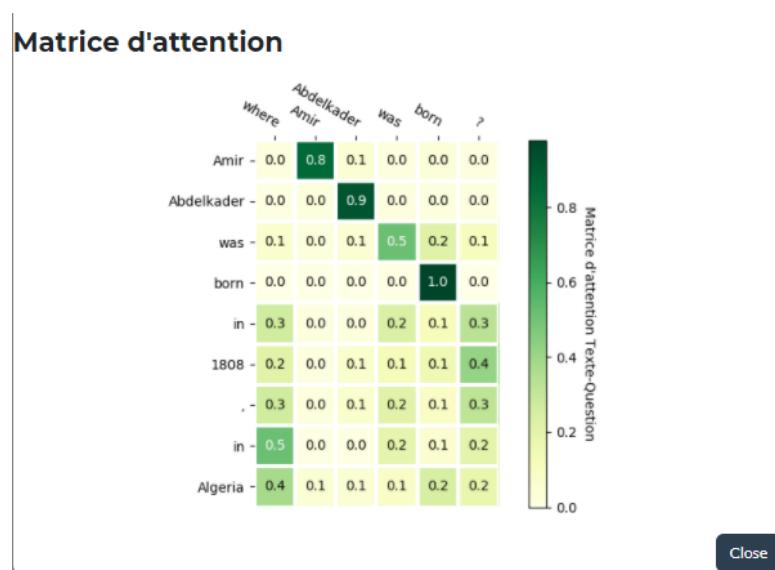


FIGURE 3.10 – Affichage d'une matrice d'attention sur l'interface

**L'affichage de tableau des distances** Pour le modèle basé sur le calcul des distances, un tableau contenant chaque phrase du texte avec sa distance par rapport à la question est affiché.

**Tableau des distances**

Phrase	Distance
Philosophers in antiquity used the concept of force in the study of stationary and moving objects and simple machines, but thinkers such as Aristotle and Archimedes retained fundamental errors in understanding force.	0.6411215960979462
In part this was due to an incomplete understanding of the sometimes non-obvious force of friction, and a consequently inadequate view of the nature of natural motion.	0.7504749447107315
A fundamental error was the belief that a force is required to maintain motion, even at a constant velocity.	0.7219461500644684
Most of the previous misunderstandings about motion and force were eventually corrected by Galileo Galilei and Sir Isaac Newton.	0.5245113372802734
With his mathematical insight, Sir Isaac Newton formulated laws of motion that were not improved-on for nearly three hundred years.	0.5903131663799286
By the early 20th century, Einstein developed a theory of relativity that correctly predicted the action of forces on objects with increasing momenta near the speed of light, and also provided insight into the forces produced by gravitation and inertia.	0.6880257427692413

FIGURE 3.11 – Affichage du tableau des distances du modèle de distances

# CONCLUSION

Dans ce travail nous nous sommes intéressés au problème connu sous l'appellation : *Question Answering* et qui consiste à déterminer des réponses à des questions posées par des systèmes intelligents.

Nous avons considéré le cas des questions factuelles et utilisé le corpus de textes SQuAD lors de l'apprentissage des modèles.

Nous avons proposé trois modèles pour solution, basés sur des architectures de Deep learning qui extraient les réponses à partir du texte :

1. Un modèle de distance.
2. Un modèle basé LSTM en deux versions : l'une sans mécanisme d'attention et la seconde, renforcé par ce mécanisme.
3. Un modèle d'apprentissage par transfert.

Les résultats des nos expérimentations permettent affirmer, sans équivoque, que le modèle d'apprentissage par transfert de BERT est le plus précis et le plus rapide à s'entraîner et à répondre avec un résultat d'exact match de 75% et F1-score de 77%. Ce résultat est dû en grande partie à son mécanisme d'attention (self attention) qui calcule l'importance de chaque mot du texte et de la question par rapport à lui-même et aux autres mots (du texte et de la question encore) sur plusieurs couches.

Un autre avantage de ce modèle est qu'il ne dispose pas de réseaux de neurones récurrents (RNN, LSTM, GRU ..) qui sont connus pour avoir une certaine lenteur dans les calculs.

Signalons néanmoins un inconvénient de ce modèle qui est le suivant : si le texte est trop grand la matrice d'attention peut être très grande et peut coûter en mémoire étant donné que c'est une matrice de taille égale à (taille du texte + taille de la question).

Le modèle LSTM sans attention n'a pas réussi à extraire les réponses durant l'entraînement. Les principales raisons de ses résultats sont dues à son architecture simple qui consiste à concaténer les vecteurs des contextes des questions et ceux des textes. Il n'a aucun moyen pour détecter les termes importants dans le texte par rapport à la question et il ne fait donc que stocker les réponses des questions déjà vues. Ce modèle est rapide lors de son exécution mais dans la majorité des cas et il retourne un résultat aléatoire.

Afin de résoudre ce problème nous avons renforcé ce modèle par un mécanisme d'attention plus léger que celui de BERT avec une taille de (Taille du texte \* Taille de la question) qui est construite à partir de deux matrices (Texte-question et Question-texte) pour calculer l'importance de chaque mot du texte par rapport aux mots de la question et réciproquement.

Ce modèle est un peu moins rapide à prédire la réponse. Ceci est dû à son architecture basée sur des réseaux de neurones récurrents de type LSTM, cependant il arrive à se rapprocher du modèle précédent avec un résultat de 58% d'exact match et 70% F1-score.

Enfin, le modèle le plus long s'exécuter est, en effet, le modèle de distances. Cette lenteur est due au fait qu'il représente des phrases en vecteurs au lieu des mots ainsi que les calculs de distances entre chaque phrase avec la question, ce qui prend largement plus de temps par rapport aux autres modèles.

Néanmoins, ce modèle arrive tout de même à trouver la phrase la plus proche de la qualité et qui pourrait potentiellement contenir la réponse avec une estimation de résultat entre 50% et 60% sur le corpus de test selon la fonction de distance utilisée.

Ces modèles sont exploitables à travers une application web qui permet à l'utilisateur d'entrer un texte et des questions et lire la réponse retournée.

L'amélioration de ce travail peut se faire en traitant les autres types de questions qui ne sont pas couverts par le dataset SQuAD et cela en utilisant d'autres datasets. Aussi il serait intéressant d'essayer les modèles basés sur les bases de connaissances. D'autres architectures de réseaux de neurones et d'autres mécanismes d'attention peuvent aussi améliorer la qualité des résultats.

Nous avons travaillé principalement sur l'Anglais vue la disponibilité des corpus. Nous nous intéressons dans un moyen terme à étudier les limites de ces modèles pour la langue arabe.

## BIBLIOGRAPHIE

- [1] Bert F. GREEN, Carol CHOMSKY et Kenneth LAUGHERY. *BASEBALL: AN AUTOMATIC QUESTION-ANSWERER*. Massachusetts Institute of Technology, 1961.
- [2] William A. WOODS. *Lunar rocks in natural english: explorations in natural language question answering*. Massachusetts Institute of Technology, 1977.
- [3] Joseph WEIZENBAUM. *ELIZA-A Computer Program For the Study of Natural Language Communication Between Man and Machine*. Massachusetts Institute of Technology, 1966.
- [4] Terry WINOGRAD. *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. Massachusetts Institute of Technology, 1971.
- [5] Maxime ALLARD. *What is a transformer*. 2019. URL: <https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>.
- [6] Jacob DEVLIN, Ming-Wei CHANG et Kenton LEE. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Google Ai Language, 2018.
- [7] Chaveevan PECHSIRI et Piriaykul RAPEPUN. *Developing a Why-How Question Answering system on community web boards with a causality graph including procedural knowledge*. Department of information technology, dhurakij pundit university, Thailand, 2016.
- [8] Ayu PURWARIANTI. *Statistical-based Approach for Indonesian Complex Factoid Question Decomposition*. Informatics Department, Universitas Muhammadiyah Malang, Indonesia, 2016.
- [9] Ruobing XIE et al. “FAQ-based Question Answering via Knowledge Anchors”. WeChat Group, Tencent Inc, 2019.
- [10] Jeffrey PENNINGTON, Richard SOCHER et Christopher D. MANNING. “GloVe: Global Vectors for Word Representation”. Stanford university, 2014.
- [11] *Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning*. 2018. URL : <https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90>.
- [12] James H. Martin DANIEL JURAFSKY. *Encoder-Decoder Models, Attention, and Contextual Embeddings*. 2019.
- [13] Google Brain team GOOGLE RESEARCH. “Attention Is All You Need”. 2017.

- [14] Pranav RAJPURKAR et al. “Squad: 100,000+ questions for machine comprehension of text”. In : *arXiv preprint arXiv:1606.05250* (2016).
- [15] NAYUKI. *Computing Wikipedia's internal PageRanks*. 2016. URL : <https://www.nayuki.io/page/computing-wikipedias-internal-pageranks>.
- [16] Minjoon Seo Aniruddha Kembhavi andAli Farhadi andHananneh HAJISHIRZI. “BI-DIRECTIONAL ATTENTION FLOWFOR MACHINE COMPREHENSION”. University of Washington, 2017.
- [17] Zhuosheng ZHANG et al. “SG-Net: Syntax-Guided Machine Reading Comprehension”. g, Shanghai Jiao Tong University, 2019.
- [18] Zhuosheng ZHANG et Junjie Yang andHai ZHAO. “Retrospective Reader for Machine Reading Comprehension”. Shanghai Jiao Tong University, 2020.
- [19] Zhenzhong Lan andingda Chen andSebastian GOODMAN et Kevin GIMPEL. “ALBERT: A LITE BERT FOR SELF-SUPERVISEDLEARNING OF LANGUAGE REPRESENTATIONS”. Google Research, 2020.
- [20] Dominique LAURENT, Patrick SÉGUÉLA et Sophie NÈGRE. *QA better than IR ?* Synapse Développement, 2006.
- [21] A. Chandra Obula REDDY et Dr. K. MADHAVI. *A Survey on Types of Question Answering System*. Department of Computer Science & Engineering, JNT University,India., 2017.
- [22] Jian FU, Xipeng QIU et Xuanjing HUANG. *Convolutional Deep Neural Networksfor Document-Based Question Answering*. School of Computer Science, Fudan University, China, 2016.
- [23] Tom YEH, John J. LEE et Trevor DARRELL. *Photo-based Question Answering*. MIT EECS, CSAIL-Cambridge, USA, 2008.
- [24] A. Chandra Obula REDDY et Dr. K. MADHAVI. *A Survey on Types of Question Answering System*. Research Scholar, Department of Computer Science & Engineering, India, 2017.
- [25] Ajay KAPOOR. *deep learning vs machine learning , a simple explanation*. 2018. URL : <https://hackernoon.com/deep-learning-vs-machine-learning-a-simple-explanation-47405b3eef08>.
- [26] TENSORFLOW. *Word embeddings*. URL : [https://www.tensorflow.org/tutorials/text/word\\_embeddings](https://www.tensorflow.org/tutorials/text/word_embeddings).
- [27] Raimi KARIM. *Attn: Illustrated Attention*. 2019. URL : <https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#1d00>.
- [28] Harshall LAMBA. *Intuitive Understanding of Attention Mechanism in Deep Learning*. 2019. URL : <https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f>.