Research and Methodology in Data Science

Question Answering Systems

Étudiants :
Idles MAMOU
Amine DJEGHRI

Numéros Étudiants :
3803676
3801757

Janvier 2021

Table des matières

1 Question Answering

1.1 Définition

Les systèmes questions-réponses sont des systèmes d'interaction homme-machine permettant d'extraire des informations à partir de données, structurées ou non structurées, à l'aide de requêtes formulées en langage naturel.
Ces systèmes se basent généralement sur du texte en entrée bien que certains d'entre eux acceptent également la saisie vocale, tels que Alexa d'Amazon ou Siri d'Apple, et produisent une réponse concise en sortie.
Le moteur de recherche de Google ajoute une forme de réponse aux questions en plus de ses résultats de recherche traditionnels, comme illustré dans la figure ci-dessous :

FIGURE 1.1.1 – Résultat d'une question sur Google

Domaine des Systèmes Questions-Réponses
Les systèmes questions-réponses fonctionnent dans un domaine, contraints par les données qui leur sont fournies. Le domaine représente l'incarnation de toutes les connaissances que le système peut connaître. Il existe deux paradigmes de domaine : ouvert et fermé. Les systèmes à domaine fermé ont une portée étroite et se concentrent sur un sujet spécifique. Les systèmes à domaine ouvert sont vastes et répondent à des questions de connaissances générales.
Le système BASEBALL est un des premiers exemples de système à domaine fermé. Construit dans les années 1960, il se limitait à répondre à des questions sur une année de faits et de statistiques sur le baseball. Non seulement ce domaine était limité au sujet du baseball, mais il était également limité en terme de quantité de données à portée de main. Un exemple contemporain de systèmes à domaine fermé est celui de certaines applications BI. En règle générale, leur domaine est limité aux données fournies par l'utilisateur, de sorte qu'il ne peut répondre qu'aux questions sur les ensembles de données spécifiques auxquels il a accès.
En revanche, les systèmes questions-réponses à domaine ouvert s'appuient sur des connaissances fournies par de vastes ressources - telles que Wikipedia ou le Web - pour répondre à des questions de connaissances

générales. Un exemple d'un tel système est le moteur de recherche de Google.

1.2 Historique

Les premiers travaux sur les systèmes questions-réponses reposaient principalement sur des règles syntaxiques conçues manuellement pour répondre à des requêtes en raison de ressources informatiques limitées, comme Baseball en 1961, Lunar en 1977 et Janus en 1989. Vers l'an 2000, plusieurs conférences telles que TREC-QA et QA@CLEF, ont largement promu le développement du Question Answering. Un grand nombre de systèmes qui utilisent des techniques de recherche d'information (RI) ont été proposé au début du 21ème siècle. Puis vers 2007, avec le développement de bases de connaissances (KB), comme *Freebase* ou *DBpedia*, et l'émergence de jeux de données à domaine ouvert comme *WebQuestions* et *SimpleQuestions*, les technologies dites "Knowledge Based QA" ont pris de l'ampleur. En 2011, IBM Watson, un modèle hybride qui mélange les techniques de RI et de KBQA, a remporté le jeu télévisé "The Jeopardy !". A partir de 2015, en raison de la publication de plusieurs datasets de référence à grande échelle et du développement rapide du Deep Learning, de grands progrès ont été réalisés dans le domaine de la compréhension de texte (Machine Reading Comprehension), ce qui a eu un impact conséquent sur le Question Answering.

2 Types de Systèmes Questions-Réponses :

Selon le type de données sur lesquelles les systèmes QA se basent, ces derniers peuvent être catégorisés en deux types :

2.1 Modèles à base de Graphes de Connaissances (Knowledge Based QA) :

Dans les approches de ce type, les données en grande quantité sont sauvegardées sous une forme structurée, ex : des bases de données relationnelles (DB) ou des bases de données structurées plus simples comme des ensembles de triplets RDF (KB). L'objectif des systèmes Knowledge Based est de traduire les requêtes écrites en langage naturel dans le vocabulaire de ces entités structurées grâce à des algorithmes d'analyse sémantique. Ces techniques d'analyse sémantique convertissent les chaînes de texte en logique symbolique ou en langages de requête, comme le SQL ou SPARQL.

2.1.1 Les Dataset

Avec le développement en 2008 de la base de connaissance à large échelle *FreeBase*, plusieurs ensembles de données (Tableau 2.1.1) ont vu le jour :

WebQuestions : introduit en 2013 par *Bearnt et al* et basé sur *FreeBase*, ce benchmark est beaucoup plus large (5 810 questions) et plus bruité que les précédents utilisés jusqu'à lors. Les questions sont également plus réalistes (elles ont été obtenus via l'API Google) et plus diversifiées sur le plan linguistique (Beaucoup plus de prédicats). Il a cependant deux principaux défauts : (i) seulement des paires de questions réponses en langage naturel sont fournies, aucune forme logique n'est fournie. (ii) 84% des questions sont relativement simples et ne requiert pas de contrainte d'inférence

ComplexQuestions : introduit en 2016 par *Bao et al*, ce dataset, comme *WebQuestions* est basé sur *FreeBase*. Pour palier au problème de la simplicité des questions, ce dataset fourni prés de 2100 questions qui implique des contraintes de temps ou d'agrégation.

WebQuestionsSP : introduit en 2018 et conçu pour les questions relativement simples, ce dataset vient principalement apporter une réponse au premier problème. Il contient 4 737 exemples et est fourni avec en

plus des paires de questions-réponses la traduction sous forme logiques des questions.

ComplexWebQuestions : introduit en 2018 et basé sur *FreeBase*, il est conçu pour des questions hautement complexes et contient 34 689 exemples (contre 2100 pour CompQ). Il a été conçu en modifiant les requêtes SPARQL du *WebQuestionsSP* en y incluant beaucoup plus de contraintes.

Dataset	Background KB	Size	Logical forms
WebQuestions [1]	Freebase	5,810	No
ComplexQuestions [2]	Freebase	2,100	No
WebQuestionsSP [3]	Freebase	4,737	Yes
ComplexWebQuestions [4]	Freebase	34,689	Yes

FIGURE 2.1.1 – Les différents benchmark du Knowledge Based Question Answering

A partir de 2016, d'autres ensembles, conçus notamment sur la base de connaissances *DBpedia*, ont aussi vu le jour.

On distingue deux principales approches Knowledge Based :

2.1.2 Approches Traditionnelles

Les méthodes traditionnelles des approches Knowledge Based reposent principalement sur des règles ou des modèles (templates) prédéfinis à la main pour analyser sémantiquement les questions et obtenir des leur traduction sous forme logique.

Avec l'émergence de la base *FreeBase*, le principal défi est de réduire le grand nombre de prédicats logiques possibles pour une question donnée. Pour palier à ce problème, *Berant et al* ont proposé en 2013 un analyseur sémantique qui fonctionne en deux étapes comme suit : (i) Tout d'abord, un mapping des mots en langage naturel vers des prédicats est construit en utilisant la base de connaissances et un grand corpus de textes. (ii) Puis, étant donné une question, l'analyseur génère de manière récursive des prédicats candidats en se basant sur les prédicats des mots voisins dans la question grâce au lexique généré en (i) (Figure 2.1.2). Au final, l'analyseur s'appuie sur un modèle linéaire et des features construits à la main pour calculer le score de chacun des candidats.

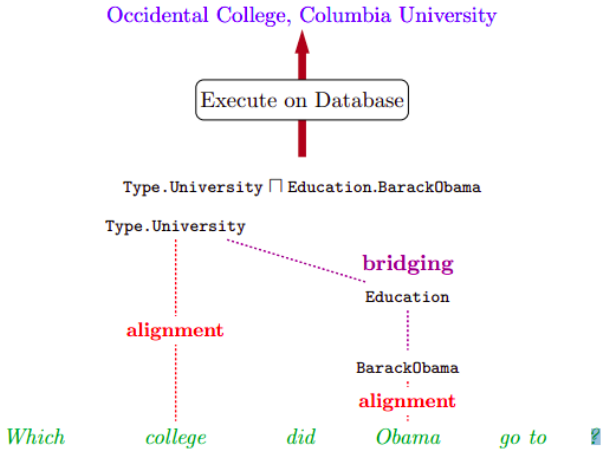

FIGURE 2.1.2 – Analyseur Sémantique Berant et Al (2013)

Bien que le dataset *WebQuestions* soit plus large, plus diversifié et plus bruité que les précédents dataset, le modèle est capable d'atteindre un F1 score moyen de 31,4 %, une amélioration de 4,5 % par rapport à une baseline naturelle. Cependant, ce modèle dépend beaucoup d'une main experte pour créer les lexiques (mapping).

Pour palier à la dépendance du modèle précédant vis à vis de la qualité du lexique construit, *Bast et al* ont proposé en 2015 un modèle basé sur des templates, AQQU, qui cartographie les questions en trois types comme le montre la figure 2.1.3.
Étant donné une question, toutes les entités qui correspondent à un des templates sont identifiées à partir de la base de connaissances. Ensuite, AQQU utilise la base de connaissance pour construire le sous-graphe centré sur la ou les entités candidates. Pour chaque question, les candidats à la requête sont classés suivant le score donné par un modèle de ranking qui prend en entrée des vecteurs de caractéristiques (définis à la main) pour chaque candidat. La requête la mieux classée est ensuite utilisée pour fournir le réponse.

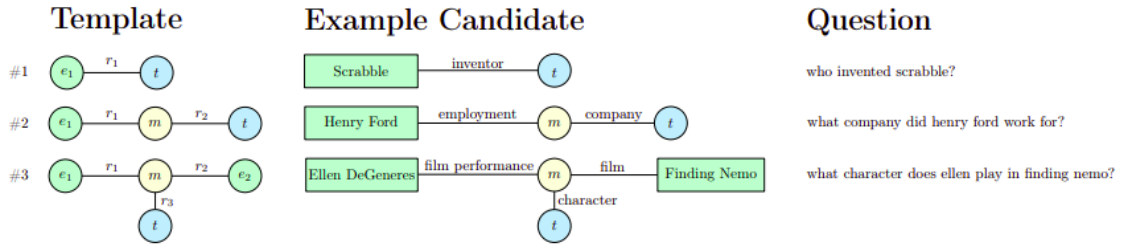

FIGURE 2.1.3 – Les 3 types de de questions et les exemples candidats correspondants (Bast et al 2015)

Ce modèle atteint des performances en test de très bonne qualité (49.4 % de moyenne F1 score sur le benchmark *WebQuestions*).Cependant, seulement 3 types (templates) de questions sont utilisés ce qui permet au modèle d'être performant que sur des benchmark de questions assez simples (comme WebQuestions). Ce modèle n'offre donc pas la possibilité de répondre aux questions complexes.

2.1.3 Approches à base de Réseaux de Neurones

Contrairement aux méthodes traditionnelles qui s'appuient des règles ou templates, ces approches tentent de réduire au maximum la part d'expertise humaine dans les modèles en construisant des analyseurs sémantiques à base de réseaux de neurones.

Pour palier aux problèmes de généralisation des modèles traditionnelles liés à l'intervention humaine pour créer les lexiques et les templates, *Bordes et al* ont introduit en 2014 une méthode Knowledge Based basée sur l'apprentissage de représentations (Embeddings). Le modèle fonctionne en apprenant les projections vectorielles de faible dimension des mots apparaissant dans les questions d'une part, et des réponses candidates représentées par la somme des vecteurs de trois aspects : l'entité de réponse, le chemin de relation entre l'entité de réponse et l'entité de sujet dans la base de connaissance, et le sous-graphe lié à l'entité de réponse.(Figure 2.1.4)
La pertinence sémantique d'une réponse vis à vis d'une question est calculée par produit scalaire des projections, et une margin-loss est utilisée pour entraîner les paramètres du modèle.

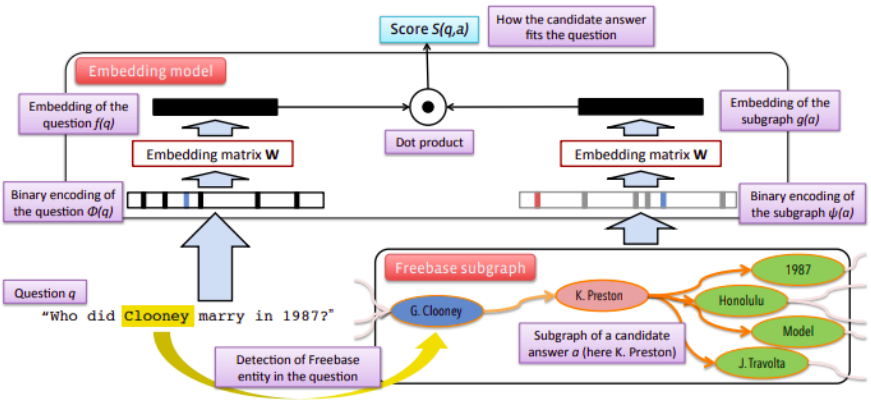

FIGURE 2.1.4 – Architecture du modèle à base d'Embeddings (Bordes et al 2014)

En terme de performance, le modèle atteint un F1 score de 39.2 % sur le benchmark *WebQuestions*. C'est mieux que l'analyseur à base de lexique (Berant et al) mais moins que le modèle AQQU à base de templates.

En 2015, *Yih et al* ont proposé un analyseur sémantique, STAGG, qui exploite d'avantages et plus tôt dans le processus la base de connaissances. Concrètement, l'analyse sémantique a pour but la génération d'un graphe de requêtes représentant un sous graphe du graphe de connaissances de la base. L'exploitation de ce graphe a pour effet la réduction de l'espace de recherche simplifiant ainsi le problème de correspondance sémantique. STAGG définit trois étapes pour générer des graphiques de requête. (i) Tout d'abord, un extracteur d'entités *(Yang and Chang, 2015)* est utilisé pour obtenir les entités principales dans la question et qui seront le point de départ des graphes (Figure 2.1.5). (ii) Ensuite, pour chaque graphe, STAGG explore tout les chemins de relation entre l'entité extraite dans la première étape et le nœud de réponse. Pour identifier le chemin le plus pertinent jusqu'à la réponse, un réseau de convolution CNN est utilisé pour scorer chaque chemin (comme pour Bordes et al)(Figure 2.1.6). (iii) Finalement, des nœuds de contrainte, soit d'autres entités ou une formule d'agrégation (ex : Minimum), sont attachés au chemin retenu par le CNN selon des règles heuristiques(Figure 2.1.7). Une fois les graphes générés, un modèle de ranking (ex : lambdaRank) est utilisé pour obtenir le meilleur graphique de requête (Figure 2.1.8) qui sera utilisé pour interroger la base.

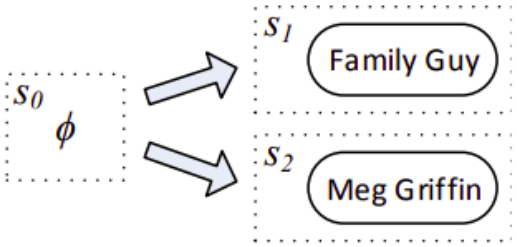

FIGURE 2.1.5 – Etape 1 : Extraction des entités principales dans la requête

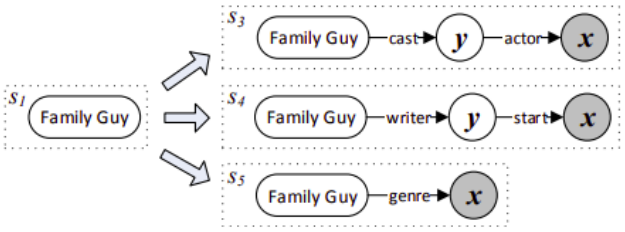

FIGURE 2.1.6 – Étape 2 : Identification du meilleur chemin dans le graphe à l'aide de CNN

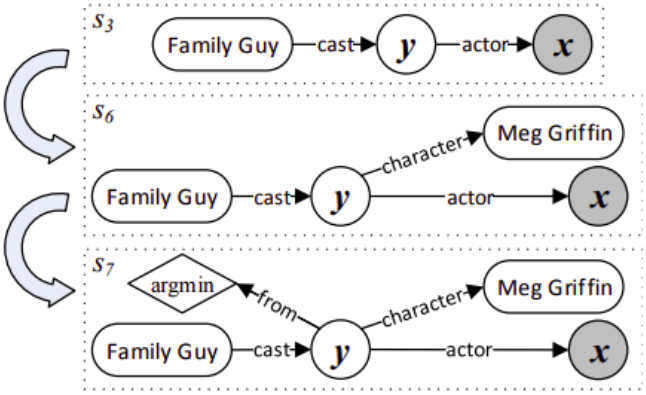

FIGURE 2.1.7 – Étape 3 : Ajout des contraintes

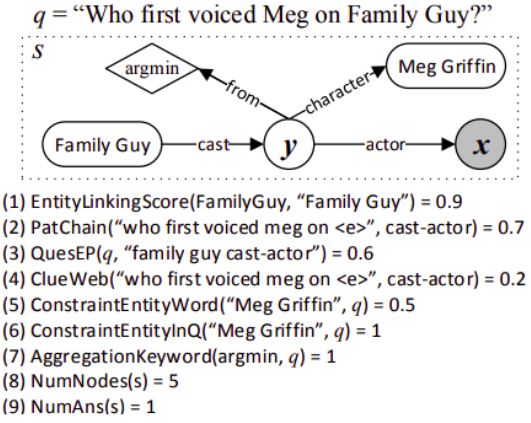

FIGURE 2.1.8 – Les différents features pour l'étape de ranking des graphes

Avec 52.5% de F1 score sur le benchmark *WebQuestions*, le STAGG est plus performant que les modèles cités plus haut. Cependant, lorsqu'on teste sur des datasets comme *ComplexQuestion* contenant des questions plus complexes, qui renvoient à plusieurs entités et à de multiples relations entre elles, on observe une chute de F1 score à 37%.

Dans la période qui a suivi, beaucoup d'améliorations du STAGG ont été proposées pour répondre notamment aux questions complexes.

Bao et al, en 2016, ont étendu les types de contraintes et les opérateurs basés sur STAGG, y compris les contraintes de type et les contraintes de temps explicites et implicites, et ont proposé un graphe de requête à contraintes multiples (MultiCG) pour résoudre les questions complexes. Le modèle obtient en test un F1 score de 42.3% sur la base *ComplexQuestions* (contre 37% pour le STAGG) et 54% sur *WebQuestions*.

Pour restreindre l'espace de recherche, STAGG explore uniquement les chemins de relation de longueur 2 au maximum, il est donc impossible de traiter les questions à sauts multiples. Certains chercheurs ont donc proposé un cadre de génération de graphes de requête itératif.
En 2019, *Bhutani et al* ont proposé le modèle TEXTRAY selon l'hypothèse qu'un graphe de requête complexe peut être construit par composition de plusieurs graphes de requêtes simples.
Pour chaque requête partielle, des chemins candidats sont construits par STAGG et un score de similarité sémantique avec la question est calculé par un modèle neuronale à base de LSTM et d'Attention (Figure 2.1.10). Les k meilleurs candidats pour chaque requête partielle sont alors exécutés, leurs réponses (entités) aident à trouver les candidats pour la requête suivante et ainsi de suite. De cette façon, plusieurs dérivations de requêtes complètes sont générées à partir d'une requête simple. La dérivation avec le score global le plus élevé est finalement exécutée. L'architecture du modèle est décrite dans la figure 2.1.9.

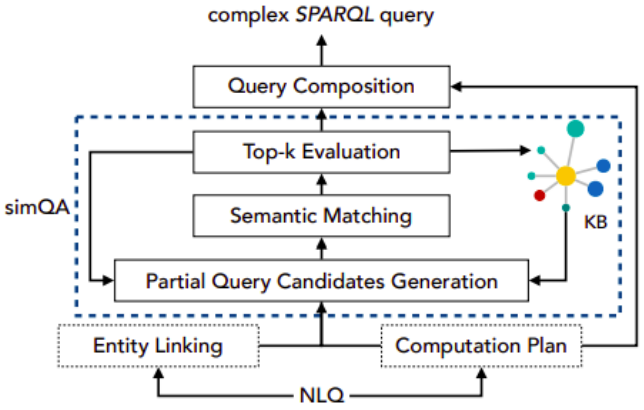

FIGURE 2.1.9 – L'architecture globale du modèle TextRay

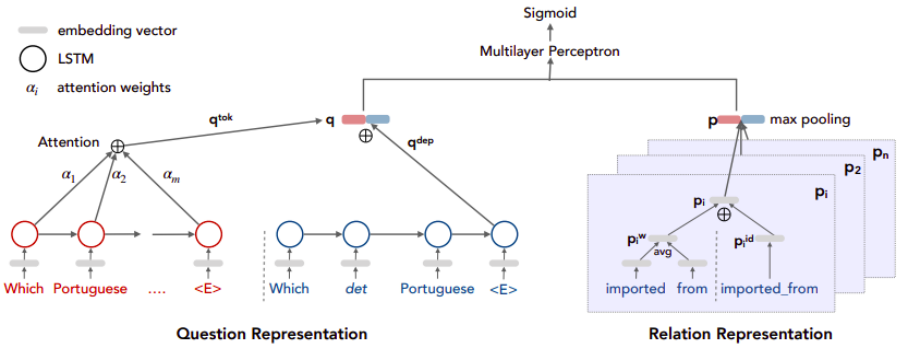

FIGURE 2.1.10 – L'architecture du modèle de similarité sémantique

De leur coté, *Lan et al* ont proposé en 2020 de re-visiter le modèle STAGG d'une manière plus flexible permettant de prendre en considération des chemins candidats plus longs (supérieurs à 2 sauts) tout en réduisant leur nombre. Ils ont donc proposé un STAGG qui n'attend pas que chaque chemin de relation de base soit généré complètement pour y attacher une contrainte(Etape 3 du STAGG). Cette manière plus flexible de générer des graphiques de requêtes, associée à un mécanisme de Beam Search explore un bien plus petit espace de recherche tout en conservant une chance élevée de trouver le meilleur graphe de requête. Pour l'étape de ranking finale (Après génération des différents graphes), on donne en entrée au lamdaRank un vecteur de features en 7 dimensions dont la première provient d'un score de similarité sémantique basé sur une représentation BERT.

En terme de performance, ce modèle bat les principaux modèles de l'état de l'art sur les 3 benchmark du Knowledge Based Question Answering : *ComplexQ, ComplexWebQuestions, WebQuestionsSP* (Tableau 1).

	ComplexWeb-Questions	**WebQuestionsSP**	**ComplexQuestions**
STAGG (Yih et al, 2015)	-	69.0	37.0
MultiCG (Bao et al, 2016)	-	-	42.3
Luo et al (2018)	-	-	42.8
Chen et al (2019)	29.8	68.5	35.3
TextRay (Buhtani et al, 2019)	33.9	60.3	-
Lan et al (2020)	40.4	74.0	43.3

TABLE 1 – F1 score (%) des différents modèles de l'état de l'art du Knowledge Based QA

Les approches Knowledge Based sont très performantes sur des questions simples et les dernières avancées dans ce domaine améliore considérablement les performances sur les questions complexes. Cependant, ces approches restent très dépendantes des ontologies des bases de connaissances. A cela s'ajoute la nécessité de maintenir tout le temps à jour ces bases de connaissances (ex FreeBase).
Pour éviter ces problèmes liés aux bases de connaissances, une autre famille de modèles tente de répondre à la problématique de l'Open Domain Question Answering par des méthodes de recherche d'information (RI) et de compréhension automatique de texte (MRC), ce sont les approches IR Based.

2.2 Modèles à base de Recherche d'information (IR Based QA)

Dans ce type d'approches, contrairement aux approches Knowledge Based qui extraient les réponses depuis des bases de connaissances (données structurées), les modèles tentent d'extraire des réponses en travaillant directement sur de grandes bases de documents comme Wikipedia ou le Web (données non structurées). Pour ce faire, des outils de recherche d'informations sont d'abord utilisés pour filtrer les paragraphes ou les documents dits "candidats" susceptibles de contenir la réponse, puis des techniques de compréhension de texte sont utilisées pour en extraire la réponse. Les dernières avancées dans ce domaine sont donc fortement liées aux avancées observées dans les domaines de la RI et du *Machine Reading Comprehension* MRC.

2.2.1 Les Dataset

A l'inverse des modèles de MRC pour lesquels les paragraphes contenant la réponse sont fournis avec les dataset, les modèles IR Based ont pour but de retrouver les réponses aux questions dans de larges collections de documents (ex Wikipédia). Les benchmark pour ces approches sont donc composés seulement de paires questions-réponses et ne contiennent pas les paragraphes.

SQuAD-open : introduit en 2017 par *Chen et al*, ce dataset d'Open Domain QA est basé sur le dataset de compréhension de textes SQuAD (2016) qui contient 100k questions-réponses construites en se basant sur Wikipédia. Dans cette version, seules les paires de questions-réponses sont fournies.

SearchQA : introduit en 2017 par *Dunn et al*, ce dataset contient 140k paires construites à l'aide du moteur de recherche Google.

TriviaQA-unfiltred : introduit en 2017 par *Joshi et al*, il contient plus de 110k paires de questions-réponses obtenues sur sites d'anecdotes.

Quasar-T : introduit en 2017 par *Dhingra et al*,il se compose principalement de 43k questions et leur réponses obtenues à partir de diverses sources Internet.

Les caractéristiques de ces différents benchmark sont décrites dans le tableau suivant :

dataset	query form	answer form	question source	context source	granularity
SQuAD-open [37]	full question	span	crowdsourced	Wikipedia	document level
SearchQA [15]	key word/ phrase	span	Jeopardy!	Google search results	paragraph level
TriviaQA [14]	full question	span	Trivia websites	Wikipedia & Bing search results	document level
Quasar-T [30]	full question	span	Free Database	CluWeb09	paragraph level

FIGURE 2.2.1 – Les benchmarks de l'Open Domain Question Answering

2.2.2 Approches IR Based

La figure 2.2.2 présente l'architecture générique des modèles IR Based. D'abord, Le module de RI se charge d'indexer et de retrouver les documents les plus pertinents pour une donnée. Le module de compréhension extrait ensuite plusieurs réponses candidates grâce à des techniques de MRC. Enfin, le système choisit la prédiction la plus prometteuse comme réponse. Accessoirement, pour booster la vitesse de traitement tout en assurant la précision, plusieurs techniques d'accélération sont adoptées.

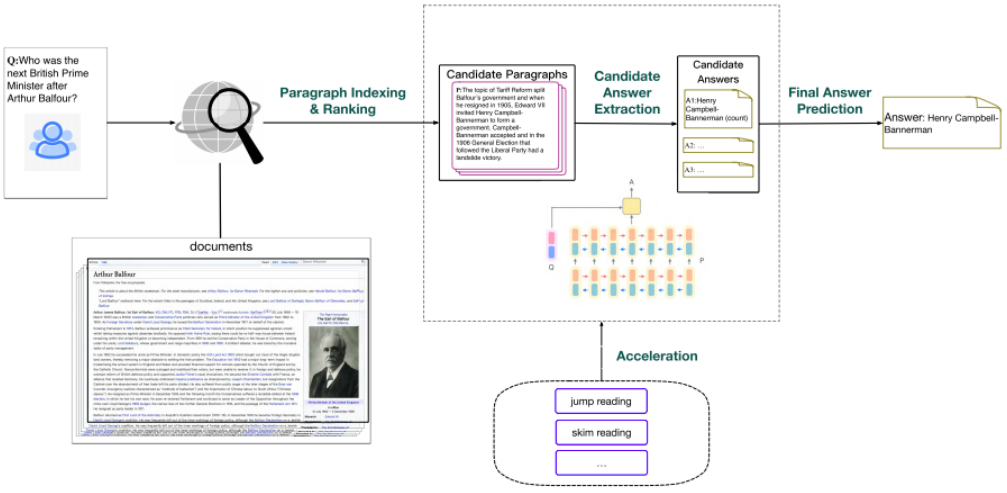

FIGURE 2.2.2 – L'architecture générique des modèles IR based

En 2017, *Chen et al* ont proposé le système DrQA (Figure2.2.3) qui a pour objectif de répondre aux questions en se basant uniquement sur les documents textuels de Wikipedia. Le modèle est principalement constitué de deux modules. Le premier module *"Document Retriever"* construit d'abord un index inverse à base de TF-IDF sur des bi-grammes. Un appareillement par similarité cosinus est ensuite effectué afin de retrouver les 5 articles Wikipedia les plus pertinents.
Afin d'extraire les réponses des articles fournis par le Document Retriever, le module *"Document Reader"*, effectue une projection vectorielle à base de *GloVe* des mots des paragraphes et des requêtes. De la co-attention entre les tokens des paragraphes et ceux de la requête est aussi ajoutée en tant que feature supplémentaires. Deux Réseaux récurrents LSTM bidirectionnels et multicouches sont ensuite appliqués de part et d'autres pour obtenir la représentation contextuelle de la requête ainsi que des tokens des différents paragraphes. Enfin, deux modèles linéaires sont utilisés pour déterminer respectivement le token de début et le token de fin de la réponse à extraire en optimisant un critère de maximum de vraisemblance.

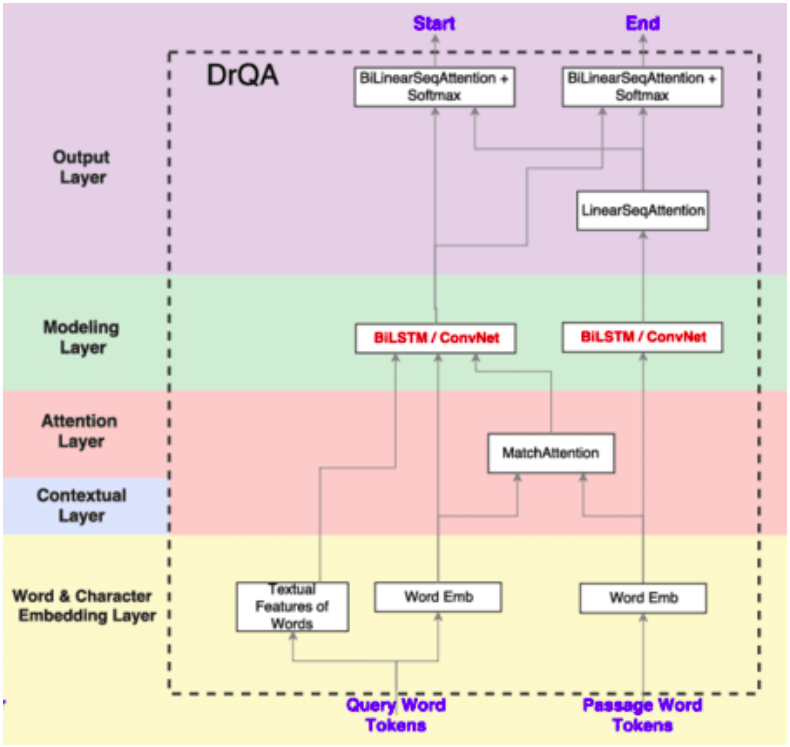

FIGURE 2.2.3 – L'architecture du modèle DrQA

En terme de performance, le DrQA obtient un score avoisinant les 28.4% d'accuracy (Exact Matching) sur le benchmark *SQuAD-open* qui traite la tache de l'Open Domain Question Answering, tandis que le *Document Reader* seul atteint, sur la tache de compréhension de texte, les 70% avoisinant les meilleures performances sur le benchmark *SQuAD*.
L'indépendance des deux modules "Retriever" et "Reader" favorise donc la propagation des erreurs liées à la non spécificité du modèle TF-IDF à la tache de Question Answering.

Pour palier aux problèmes que pose l'indépendance du Ranker et du Reader dans le modèle DrQA, *Wang et al* ont proposé en 2018 le modèle *Reinforced-Ranker-Reader* (R^3) qui permet de lier les deux modules en se basant sur de l'apprentissage par renforcement.
Comme pour le DrQA, le module de Ranking sélectionne le passage le plus susceptible de contenir la réponse et le transmet au Reader qui lit et extrait le passage en optimisant un critère de maximum de vraisemblance.

Cependant, contrairement au DrQA, le Ranker est entraîné à l'aide de l'algorithme REINFORCE *(Wiliams 1992)* avec une récompense déterminée par la qualité de l'extraction faite par le Reader vis à vis de la réponse exacte. Cela optimise le ranking en permettant de distinguer les passages léxicalement similaires à la question mais sémantiquement différents.
Par ailleurs, pour calculer le score de similarité sémantique d'un paragraphe par rapport à une requête, les projections *Glove* des deux sont données en entrée à un modèle d'appareillement Match-LSTM *(Wang and Jiang 2016)* qui, à base de réseaux LSTM bidirectionnels et de mécanismes de co-attention, fournit une représentation vectorielle jointe du paragraphe et de la requête. Une transformation non linéaire suivie d'une normalisation est ensuite appliquée pour calculer un score d'ordonnancement. Pour le Reader, le même modèle Match-LSTM est utilisé pour le calcul du score de pertinence du paragraphe vis à vis de la question, et une recompense est ensuite distribuée au Ranker en fonction de ce score.
Pour identifier le début et la fin des réponses la même méthode que le DrQA (maximum de vraisemblance) est utilisée.

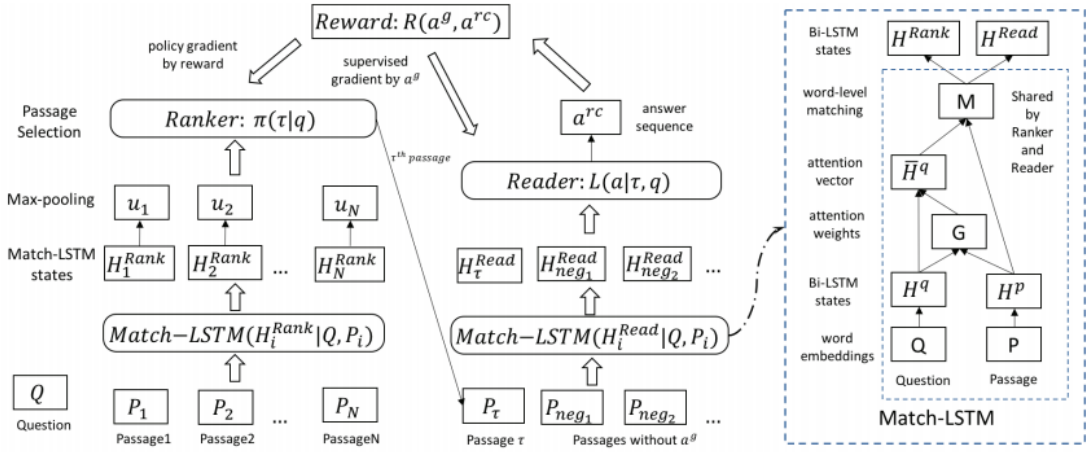

FIGURE 2.2.4 – Entraînement du modèle R^3

En terme de performances, le modèle R^3 fait légèrement mieux que le DrQA sur *SQuAD-open* (29.1% d'exact matching) mais s'incline sur les 3 autres benchmarks (Tableau 4).

En 2018, *Lin et al* ont proposé le modèle DS-QA qui reprend globalement l'idée du DrQA. Cependant pour palier aux problèmes causés par la propagation d'erreurs depuis la première étape de ranking, le DS-QA insère un module *"Paragraph Selector"* entre les deux modules de Ranking et de Reading Comprehension afin de supprimer les paragraphes issus des erreurs du Ranker.
L'intuition globale derrière le modèle est décrite dans la figure 2.2.5. Tout d'abord, un module de RI est utilisé pour retrouver les paragraphes candidats. Ces paragraphes sont ensuite donnés en entrée au *"Paragraph Selector"* qui dans un premier temps encode les mots des paragraphes et de la requête à l'aide de LSTM bidirectionnels et de la self-attention en plus pour la requête, puis calcule le score de chaque paragraphe sachant la question à l'aide d'une couche linéaire. Les paragraphes ayant les scores les plus hauts sont ensuite donnés en entrée au module Reader qui comme dans le DrQA sélectionne le début et la fin de la réponse par maximum de vraisemblance. Les paramètres du Reader et du Paragraph Selector sont optimisés par Back-Propagation.

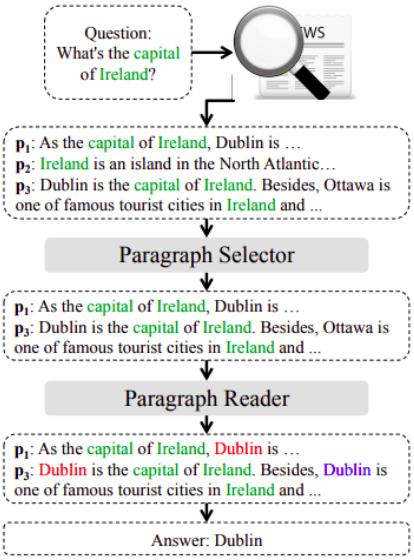

FIGURE 2.2.5 – L'architecture du modèle DSQA

En terme de performances, le DS-QA fait mieux que les modèles précédents sur 3 benchmarks (SearchQA, TriviaQ, et Quasar-T). Sur SQuAD-open, le DS-QA obtient 28.7% d'accuracy égalant la performance du DrQA.

En 2019, *Hu et al* ont proposé le modèle RE^3QA (REtrieve-REad-RErank) qui, à la différence des méthodes précédentes qui contiennent des modules Ranker et Reader séparés et un ré-encodage à l'entrée de chaque module, intègre les 3 blocks dans un modèle unifié leur permettant de partager la même représentation contextuelle. Le modèle RE^3QA est présenté sur la figure 2.2.6.
Étant donné une requête et un ensemble de documents, un filtrage TF-IDF est d'abord effectué pour diminuer l'espace de recherche. Chaque paragraphe retenu passe ensuite dans le "REtriever" qui utilise les premiers blocs de transformer de BERT pour calculer un score (par MLP et softmax) pour filtrer d'avantages les paragraphes. Les paragraphes retenus sont ensuite passés au "REader" qui utilise cette fois-ci les blocs de transformer de BERT les plus profonds pour calculer une représentation qui sera donnée à un modèle linéaire pour le calcul du score de début et de fin des passages candidats pour chacun des paragraphes. Finalement, les passages candidats redondants sont supprimés et le reste est ordonné suivant le score par le "REranker". Le modèle est entraîné de bout en bout par backpropagation.

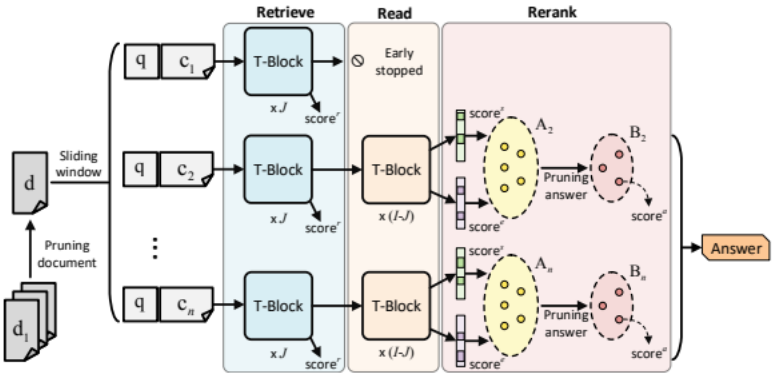

FIGURE 2.2.6 – L'architecture du modèle RE^3QA

En terme de performances, le modèle *RE*^{3}*QA* devance de loin les autres modèles avec notamment une accuracy de 41.9% sur *SQuAD-open* et 65.5% sur *TriviaQA*.

La structure des différents modèles IR Based est résumée dans le tableau 2 et leurs performances sur les différents benchmark sont résumées dans le tableau 4.

	Ranking	**Reading Comprehension**		
		Encoding	Attention	Prediction
DrQA (2017)	TF-IDF	Glove	co-attention	Bi-LSTM
R^3 (2018)	BM25+MatchLSTM	Glove	co-attention	Match-LSTM
DS-QA (2018)	TF-IDF+LSTM+MLP	Glove	self-attention	Bi-LSTM
ORQA (2019)	ICT+BERT	BERT	self-attention	BERT
RE^3*QA* (2019)	TF-IDF	BERT	self-attention	BERT
Notre Modèle	**BERT**	**BERT**	**self-attention**	**BERT**

TABLE 2 – La structure des différents modèles IR Based Question Answering

	SQuAD-open		**SearchQA**		**TriviaQA**		**QuasarT**	
	EM	F1	EM	F1	EM	F1	EM	F1
DrQA (2017)	28.4	-	51.4	58.2	48.0	52.1	36.9	45.5
R^3 (2018)	29.1	37.5	49.0	55.3	47.3	53.7	35.3	41.7
DS-QA (2018)	28.7	36.6	58.5	64.5	48.7	56.3	37.3	43.6
ORQA (2019)	20.2	-	-	-	45.1	-	-	-
RE^3*QA* (2019)	41.9	50.2	-	-	**65.5**	**71.2**	-	-
Notre Modèle	54.3	56.1	62.4	66.3	65.1	70.3	54.2	46.4
Notre Modèle (avec Transfert)	**55.2***	**57.3***	**63.5**	**67.1**	65.2	70.6	**55.6***	**47.2***

TABLE 3 – Performances des modèles IR Based (Accuracy et F1 score)

3 Nom de notre modèle

Le principal défaut des modèles de Questions-Réponses à domaine ouvert est la dépendance vis à vis des modèles de ranking utilisés pour le filtrage et qui sont à l'origine de la propagation d'erreurs vers le modèle d'extraction de réponse.
L'idée est donc de pouvoir se baser sur un modèle de ranking qui n'utiliserais pas que les correspondances lexicales pour rapprocher les passages et les questions (comme le font les modèles BM25 et TF-IDF), mais qui utiliserait aussi des informations sémantiques et contextuelles.
Ces dernières années, beaucoup de modèles de RI exploitant l'aspect sémantique des mots ont vu le jour. Ces modèles transforment généralement les requêtes et les documents en des représentations vectorielles, Pré-entraînées ou construites par le modèle lui-même, qui résument l'information sémantique des mots (DSSM, Duet,DRMM..). Plus récemment, avec le succès qu'ont eu les représentations à base de contexte (ElMo, GPT, BERT..) beaucoup de modèles basés sur ces représentations, notamment BERT, ont vu le jour impliquant une amélioration de l'état de l'art en RI. Cependant, l'application de ces modèles sur de larges corpus de documents est très coûteux en temps de calcul ce qui fait qu'en pratique ils sont utilisés que pour le ranking de documents préalablement filtrés par un premier modèle TF-IDF ou BM25.

En juin 2020, *Khattab et Zaharia* ont proposé le modèle ColBERT qui se base sur des représentations BERT des requêtes et des documents et qui introduit un mécanisme d'interaction tardive qui retarde au maximum l'interaction des deux parties. En encodant les deux parties (document et requête) indépendamment et en utilisant une couche d'interaction simple (MaxSimilarité), le modèle permet d'encoder les documents au préalable (de manière offline) ce qui permet un gain considérable en matière de coût tout en affichant de bonnes performances de ranking.

D'autres parts, *Johnson et al* ont proposé en 2017 une librairie open source FAISS (Facebook AI Similarity Search) qui permet de faire des calculs de similarité entre vecteurs en un temps recors. FAISS utilise une approximation des plus proches voisins basée sur des techniques de compression, de partitionnement et d'indexation vectorielles tirant partie du parallélisme offert par les GPU pour rendre les recherches de similarité plus efficaces.

Dans la suite, nous proposons de nous appuyer sur ces deux travaux pour nous affranchir de la dépendance vis à vis des modèles de correspondance lexicales et permettre à des passages plus pertinents d'atteindre le modèle d'extraction de réponses.

3.1 Architecture du modèle

Le Module de Ranking :

Comme nous l'avons notifié plus haut, le but de notre approche est principalement d'offrir une alternative aux modèles de ranking utilisés jusqu'à lors dans les modèles Questions-Réponses qui allie à la fois performance et moindre coût de calcul.

Pour ce faire, nous reprenons globalement le mécanisme d'interaction tardive introduit par *Khattab et Zaharia* dans ColBERT. Ainsi, nous utilisons un encodeur qui transforme chaque mot des passages ou de la requête en des vecteurs en d dimensions. L'utilisation de BERT comme encodage permet de synthétiser l'information sémantique et contextuelle de chaque mot par rapport aux autres. Le contexte globale du texte est quand à lui représenté par le vecteur du token de classification [CLS]. Cet encodage séparé des passages et des questions retarde l'interaction des deux parties et permet l'encodage des passages en amont de manière totalement indépendante des questions ce qui réduit nettement le coût de calcul lors de l'inférence **??**.

Entraînement :
Le processus d'entraînement du modèle est décrit dans sur la figure 3.1.2. Chaque exemple d'entraînement est constitué d'une question, d'un passage positif contenant la réponse ainsi qu'un passage négatif (ou plusieurs) ne contenant pas la réponse. Pour construire ces exemples, nous nous basons sur les dataset de Machine Reading Comprehension qui fournissent pour chaque question le passage contenant la réponse. Les passages négatif sont quant à eux sélectionnés parmi les passages positifs des autres questions (Ils contiennent les réponses aux autres questions). Une fois le dataset construit, chaque partie des exemples est ensuite donnée en entrée à l'encodeur qui fourni en sortie les représentations contextuelles des différents tokens. Un score de similarité cosinus est ensuite calculé entre chaque vecteur [CLS] du passage et le vecteur [CLS] de la question. Le modèle affine les paramètres de l'encodeur BERT en cherchant à minimiser la formule de coût suivante :

$$Loss(q_i, p_i^+, p_i^-) = -log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + e^{sim(q_i, p_i^-)}}$$

Où q_i représente la ième question, p_i^+ et p_i^- représente respectivement les passages positifs et négatifs.

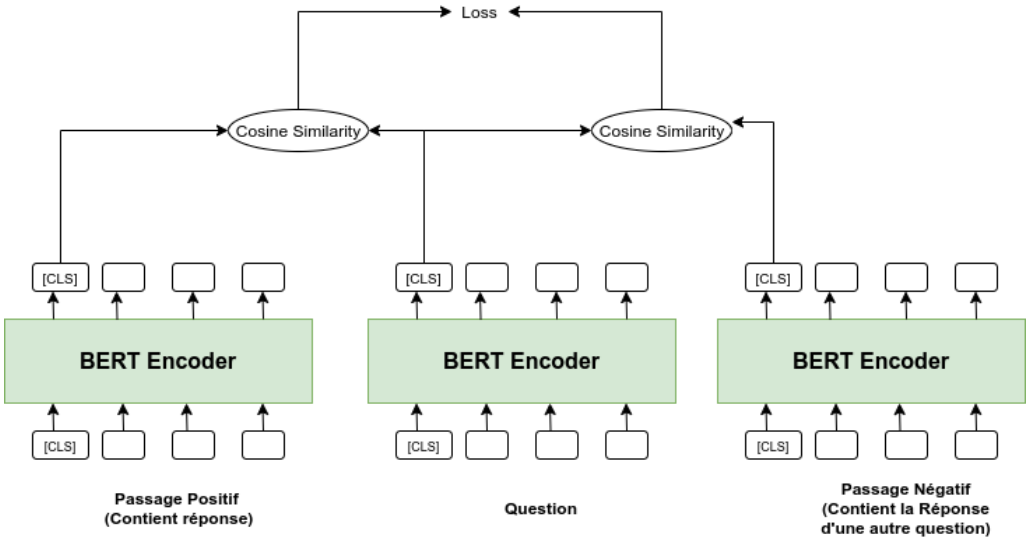

FIGURE 3.1.1 – Entraînement du Modèle

Inférence :

Comme nous l'avons notifié plus haut, le mécanisme d'interaction tardive permet d'encoder les passage/documents indépendamment des questions. L'encodage des documents peut donc se faire en amont de l'étape d'inférence. Les vecteurs obtenus sont ensuite utilisés pour la construction de l'index FAISS.
Lors de l'étape d'inférence, la question est d'abord encodée puis le vecteur [CLS] est passé à l'index de recherche de similarités FAISS qui est utilisé pour retrouver les k passages les plus proches parmi un large corpus. Cette opération est possible grâce aux mécanismes implémentés dans l'index et qui permettent à FAISS d'être appliqué à des milliards de vecteurs en contrepartie d'un temps de calcul très acceptable.

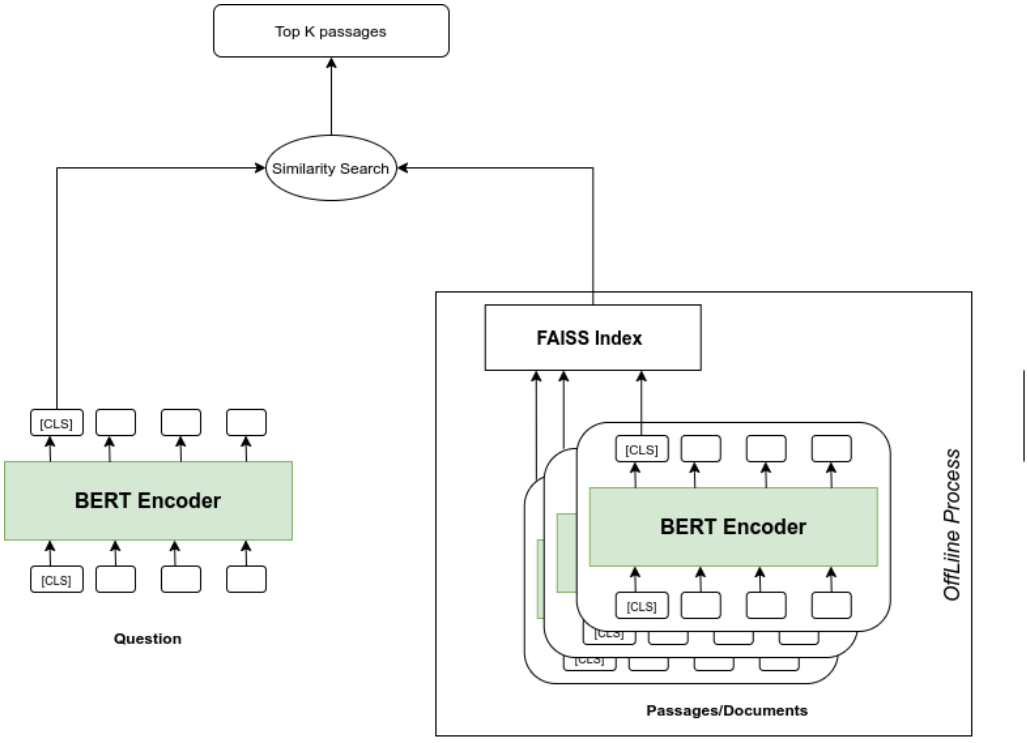

FIGURE 3.1.2 – Étape d'inférence

Le Module d'Extraction de Réponses

Une fois les passages les plus pertinents pour une question identifiés, ils sont passés au module de compréhension de texte. A ce niveau, l'objectif est de prédire l'extrait qui est le plus vraisemblable pour être la réponse. Dans ce domaine les modèles les plus récents dépassent les performances humaines sur plusieurs dataset.
Notre module reader se base comme pour le ranker sur un encodeur BERT. Plutôt que de prendre uniquement le vecteur [CLS], la représentation de chaque terme est donnée en entrée à deux modèles linéaires distincts (un pour prédire le début de l'extrait et l'autre pour la fin) et qui ont pour but de prédire pour chaque mot la probabilité qu'il soit au début ou à la fin de l'extrait :

$$P_{debut}(m) = softmax(mW_{debut})$$
$$P_{fin}(m) = softmax(mW_{fin})$$

où m est la représentation BERT du mot et W_{debut} et W_{fin} les paramètres des deux modules linéaires.

3.2 Évaluation

Modèles Knowledge Based : Les DATASET

— **WebQuestions**(2013) [**wq**] : Conçu pour les questions relativement simples.

— **ComplexQuestions**(2016) [**cq**] : Pour les questions complexes.

— Plus récemment :

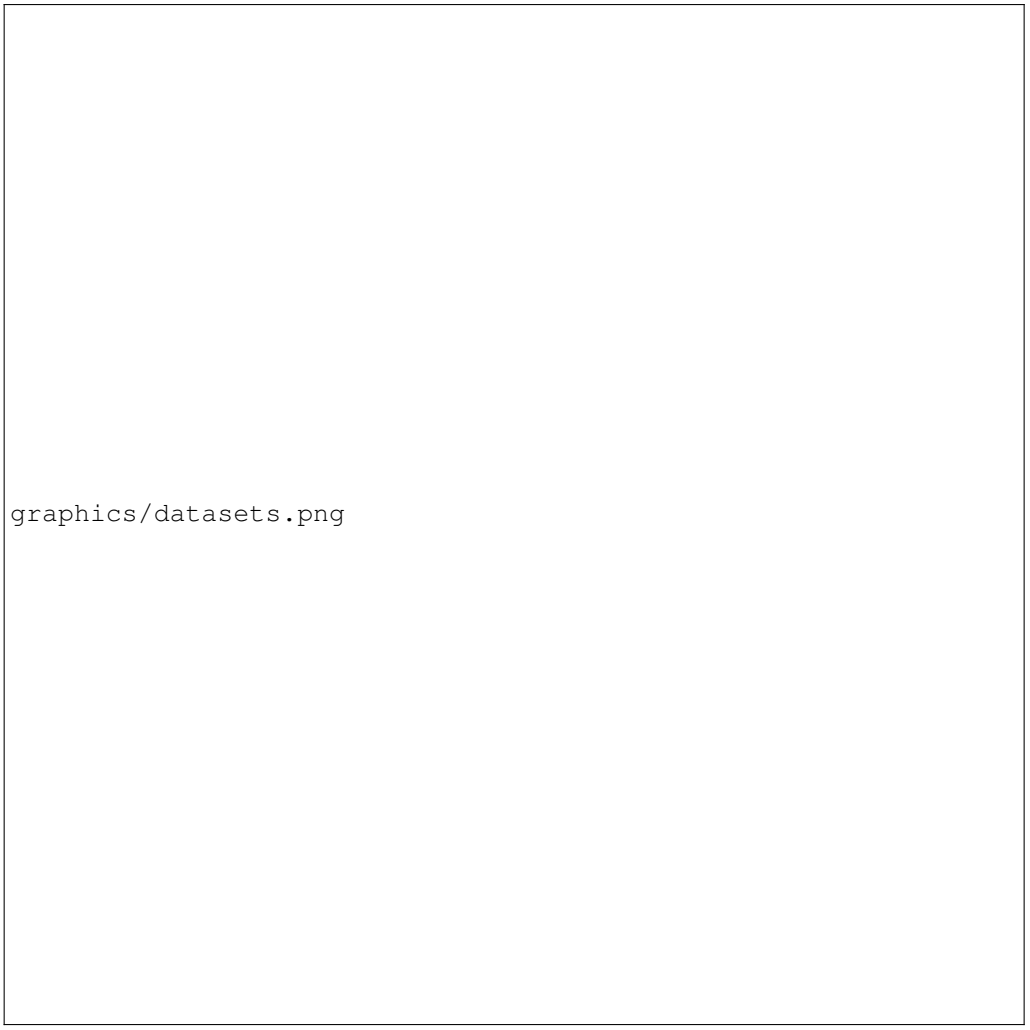

FIGURE 3.2.1 – Tableau Comparatif des Benchmarks du KB-QA

— **WebQuestionsSP**(2018)

— **ComplexWebQuestions**(2018)

Synthèse sur les Modèles IR Based

— Avantages :

1. Diminution de l'espace de recherche.
2. Plus de flexible pour les questions complexes.

— Inconvénients :

1. Dépend fortement des ontologies de la base de connaissances.

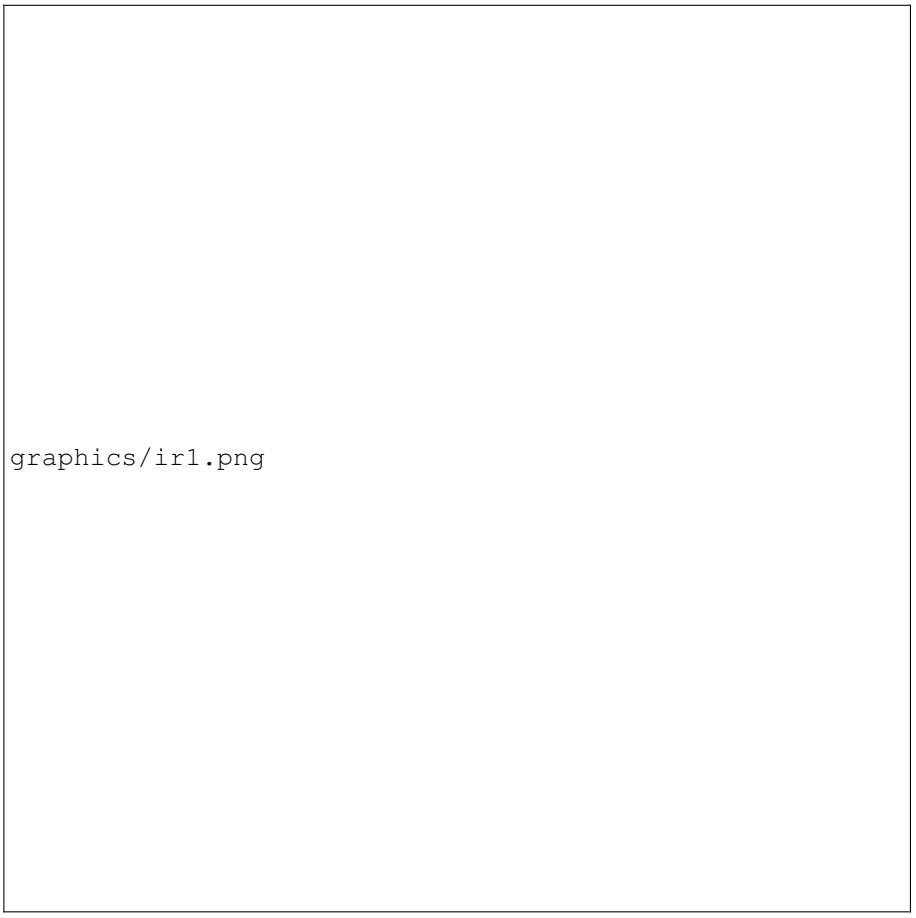

FIGURE 3.2.2 – La structure des différents modèles IR Based Question Answering

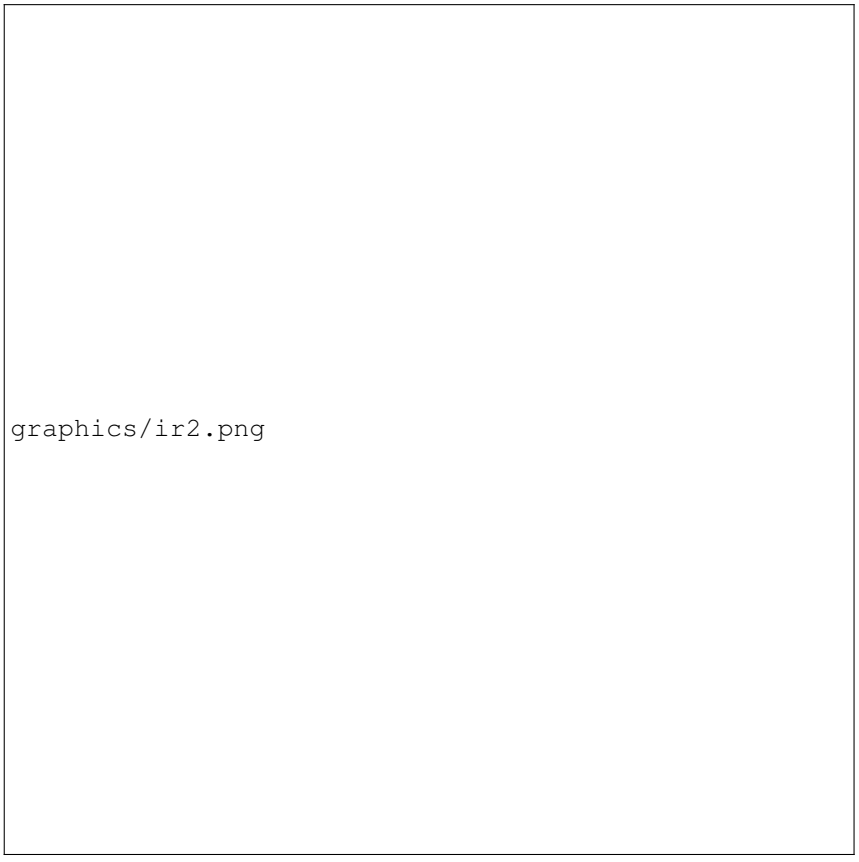

FIGURE 3.2.3 – Performances des modèles IR Based (Accuracy et F1 score) sur 4 benchmarks

	SQuAD-open		**SearchQA**					
	EM	F1	EM	F1	EM	F1	EM	F1
DrQA (2017)	28.4	-	51.4	58.2	48.0	52.1	36.9	45.5
R^3 (2018)	29.1	37.5	49.0	55.3	47.3	53.7	35.3	41.7
DS-QA (2018)	28.7	36.6	58.5	64.5	48.7	56.3	37.3	43.6
ORQA (2019)	20.2	-	-	-	45.1	-	-	-
RE^3QA (2019)	41.9	50.2	-	-	**65.5**	**71.2**	-	-
Notre Modèle	54.3	56.1	62.4	66.3	65.1	70.3	54.2	46.4
Notre Modèle (avec Transfert)	**55.2***	**57.3***	**63.5**	**67.1**	65.2	70.6	**55.6***	**47.2***

TABLE 4 – Performances des modèles IR Based (Accuracy et F1 score)