Le progrès de "Question Answering"

Zhibin Lu
zhibin.lu@umontreal.ca

Khalil Bibi
khalil.bibi@umontreal.ca

Abstract

Le "*question answering*" est un domaine de recherche qui est entrain d'évoluer avec une fréquence rapide. Ce domaine consiste à la réponse automatique sur des questions posées par des humains. Nous présentons plusieurs systèmes de QA avec quelques exemples de *data sets* utilisés pour les testes.

1 Introduction

Les tâches de "*question answering*" gagnent en importance en raison de leur applicabilité généralisée aux applications commerciales récentes telles que les chatbots, les assistants vocaux et même le diagnostic médical (Goodwin et Harabagiu, 2016). Les vieux systèmes de QA (pipeliné) utilisait le NLP afin de traiter les questions et offrir la réponse la plus probable d'être correcte à l'utilisateur. Aujourd'hui, il existe plusieurs systèmes basés sur l'apprentissage profond et d'autre systèmes à base des graphes.

Dans ce rapport nous présentons tout les systèmes listés au dessus. La section 2 s'interesse aux systèmes pipeliné (les systèmes à base de NLP et des régles). La section 3 s'interesse aux systèmes basé sur l'apprentissage profond. La section 4 va présenter les systèmes basés sur les graphes. Finalement, une petite conclusion sur tout ce qu'on a présenté.

2 Les systèmes QA pipeliné

Les recherches dans le domaine du "*question answering*" en language naturel ont commencé dès les années 1950 avec le test de turing. Le principe de ce domaine était de poser des questions factoides à des machines et de tester la performance de leurs réponses et c'est dans cette section qu'on va s'intéresser sur ce genre de question et sur les systèmes qui les traitent. Ce genre de questions était de poser une question sur n'importe quelle domaine et n'importe quel détail et c'est le rôle de la machine de parcourir le web ou les ressources disponibles (selon le système) et de fournir la meilleure réponse selon l'algorithme. Les questions étaient de genre "*When was P born ?*" ou "*When did Y happen ?*". Ensuite, ce genre de questions a évolué pour devenir de plus en plus complexe qui posait par exemple des questions bibliographiques "*Who is Barack Obama ?*", des questions pour définir des termes "*What is NLP ?*" et des questions de listes (qui ont comme réponse une liste) "*Which countries are qualified to the World Cup 2018 ?*".

Figure1 illustre un schéma de "*question answering*" :

- **Traitement des questions :** Le système fait le traitement de la question de l'utilisateur en utilisant la tokenization et en éliminant les "*stopwords*" et pour ensuite extraire les mots-clés et reformuler la question de l'utilisateur.
- **Analyse des documents :** Le système récupère les documents en relation avec la question contenant les mots-clés en utilisant un système de classement pour ces documents.
- **Analys des réponses :** Le système identifie les réponses les plus probables et affecte un score pour chacune de ces réponses. Ensuite, il identifie la réponse la plus convenable, la valide et l'envoie à l'utilisateur.

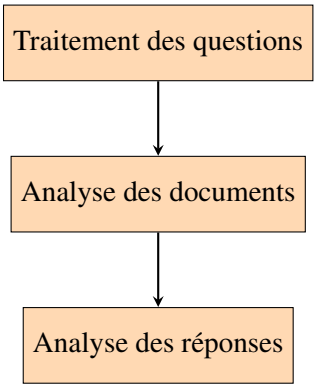

Figure 1: Schéma d'un système QA

L'évaluation des systèmes de QA a commencé en 1999, des tranches de tests pour ces systèmes ont été proposé dans "Text REtrieval Conference (TREC)" et due au succè de TREC, Cross-Language Evaluation Forum (CLEF) (Magnini et al., 2006) et NTCIR (Yutaka et al., 2005) ont proposé des tranches de tests dans plusieurs langues pour ces systèmes en se focalisant sur les langues européennes et asiatiques. Selon (Magnini et al., 2006), pour TREC et CLEF une réponse est considérée :

1. correcte si et seleuement si la réponse contient l'information requise par la question ni plus ni moins. Cette réponse doit être appuyée par le(s) document(s) dans le(s)quel(s) la réponse exacte a été trouvée et ce document doit être pertinent.

2. non prise en charge si la réponse n'est pas exacte ou le document dans lequel la réponse exacte existe n'a pas été fournie avec la réponse

3. non precise si la réponse contient plus ou moins d'information requises. Par exemple si la question demande l'année de naissance d'une personne et la réponse contient toute la date de naissance.

4. incorrecte si la réponse ne fournit pas l'information requise

Pour tester la fiabilité d'une réponse, des modèles de réponses ont été proposés. Une moyenne de 1.7 modèles par question pour les tests des systèmes TREC-8 soit 65% des questions ayant un seul modèle et 3.5 modèles par questions pour les tests des systèmes TREC-9 soit 45% des questions ayant un seul modèle de réponse. Les exemples ci-dessous illustrent des questions et des exemples de leurs modèles de réponses.

Question : *Who invented Silly Putty ?*
Modèle de réponse : General \s+Electric

Question : *Where is the location of the Orange Bowl ?*
Modèles de réponses :
^\s*Miami\s*$
Miami\s*'\s*s\s+downtown
to\s+Miami
Orange\s+Bowl\s*,\s*Miami

(David Elworthy, 2001) a essayé le système de traitement de langage naturel de Microsoft (NLPWin). NLPWin permet de convertir une question en langue naturelle en des formes logiques et des relations entre des entités (c'est la manière avec laquelle fonctionnait les systèmes avant). (David Elworthy, 2001) a montré que plusieurs types de questions ont été proposés dans les tranches de tests de TREC-8 et TREC-9 dont le tableau 1 illustre la frequence de chaque type de question (avec #TREC-8 illustre la frequence dans les tranches de tests TREC-8).

Beaucoup de systèmes ont été testé sur TREC-8 et TREC-9. (Ellen M. Voorhees, 2000) a testé ces systèmes en fournissant plus de nombre de documents pour les systèmes TREC-9 que les systèmes TREC-8 comme le montre le tableau 2.

Table 1: Fréquence des types de questions dans les tranches de tests TREC-8 et TREC-9

Type de question	#TREC-8	#TREC-9
HowDo	1	5
HowFar	1	1
HowLong	1	1
HowMany	15	27
HowManyTimes	1	0
HowMuch	3	4
HowProp	5	10
WhEquiv	1	16
WhPrep	3	35
WhRole	22	56
What	38	200
WhatMeas	1	8
WhatRole	6	36
WhatTime	6	13
When	18	48
Where	21	71
Who	28	53
WhoRole	20	62
Why	2	2
Bad	5	6
Unknown	2	39

Pour le TREC-9 le meilleur système a pu répondre à 65% des questions et le deuxième a pu répondre à 42% et ces résultats sont légèrement inférieurs à ceux du TREC-8 parceque les questions de TREC-9 étaient plus difficiles et le nombre des documents était presque le double pour le TREC-9. Les résultats obtenus par (Ellen M. Voorhees, 2000) pour TREC-9 ont montré que si on augmente la longueur de la réponse et on obtient de meilleurs résultats puisqu'ils sont pass'es de 58% de réponses correctes, lorsque la longueur de la réponse était limitée à 50 caractères, à 76% de réponses correctes lorsque la longueur de la réponse a été limitée à 250 caractères.

Le probème majeure des tranches de test TREC était la variance des questions, (Ellen M. Voorhees, 2000) a expliqué que cette variance est la conséquence de l'utilisation des mots synonymes donc par exemple les systèmes qui ont été testé ont eu un score supérieur lorsque la question était "*Where was Poe born ?*" par rapport aux autres scores qu'ils ont eu lorsque la question demandait la même information sur le lieu de naissance de Poe mais en suivant une autre formulation.

Table 2: Les données utilisées pour tester les systèmes par (Ellen M. Voorhees, 2000)

	TREC-8	TREC-9
Nombre de documents	528,000	979,000
Nombre de questions disponibles	200	693
Nombre de questions évaluées	198	682

3 Les systèmes QA à base de l'apprentissage profond

Dans cette section nous présentons les systèmes basé sur l'apprentissage profond. Ces dernières années, le réseau de neurones est devenu la méthode de recherche la plus importante dans le domaine de l'IA. Il a également grandement contribué au progrès de la discipline de traitement du langage naturel. Il existe également de nombreux résultats récents sur le QA dans les réseaux de neurones. Ce type de système a comme input un bout de texte et une questions en langage naturel sur le texte et il a comme output la réponse sur la question.

3.1 Cadre d'évaluation

Tout le monde aime utiliser le même cadre et l'ensemble de données pour tester différents modèles de réseaux neuronaux pour la comparaison. Beaucoup de benchmarks ont été proposé afin de tester ces systèmes tel que SQuAD et bAbI. Dans notre article nous nous concentrons sur bAbI qui a été proposé par (Weston et al., 2015) dans un cadre d'évaluation pour la compréhension et le raisonnement du texte, et donne 20 tâches d'évaluation (bAbI).

La suite de tâches bAbI se compose de 20 tâches de raisonnement qui comprennent la déduction, l'induction, la recherche de chemin, etc. Chaque exemple de tâche inclut un ensemble d'énoncés composés de plusieurs phrases, une question et une réponse, c'est-à-dire une séquence de phrases dont il y a suffisamment de détails pour répondre à la question. En outre, l'ensemble de données fournit la réponse correcte pour chaque question. Une déclaration peut être aussi courte que deux

phrases et aussi longue que 320 phrases. Pour répondre à la question, il est nécessaire de trouver une ou plusieurs phrases pertinentes à une question donnée et d'en tirer une réponse, ou déduire la réponse. La réponse est généralement un seul mot, mais dans quelques tâches, les réponses sont un ensemble de mots. Chaque tâche est considérée comme réussie lorsque la précision est supérieure à 95%. La plupart des modèles précédents testent leur précision sur un ensemble de données de 10k avec une formation conjointe.

Un exemple de tâche : La tâche illustrée dans le tableau 3 teste la compréhension du modèle du comportement de la relation entre les trois objets. L'objectif du teste c'est de voir si le système peut distinguer entre les sujets, les objets directs et les objets indirects. La réponse à cet exemple est dans la dernière phrase de la séquence, le sujet indirect de la phrase – la personne qui a reçu le lait de Jeff. Le modèle doit distinguer entre la cinquième phrase et la sixième phrase, exactement le contraire. Bien sûr, notre système ne reçoit aucune formation explicite sur ce qu'est le sujet ou l'objet, mais nous devons déduire cette compréhension à partir d'exemples de données d'apprentissage.

Un autre problème mineur que le système doit résoudre est de comprendre les différents synonymes utilisés dans l'ensemble de données. Jeff "apporte" du lait à Bill, mais il pouvait simplement "donne" ou "passe". Cependant, à cet égard, le modèle ne doit pas partir de zéro. Il a obtenu de l'aide sous la forme de la véctorisation de mots, qui peut stocker des informations sur la définition d'un mot et sa relation avec d'autres mots. Des mots similaires ont des formes vectorielles similaires, ce qui signifie que les modèles peuvent les traiter comme des mots presque identiques.

Table 3: Une exemple de tâche

Context :	Fred picked up the apple there. Bill travelled to the kitchen. Bill got the milk there. Jeff went to the kitchen. Bill passed the milk to Jeff. Jeff handed the milk to Bill.
Question :	Who did Jeff give the milk to?
Answer :	Bill

3.2 Approches neuronales

Dans cet ensemble de tests, les gens ont développé une vingtaine de modèles de réseaux neuronaux, nous nous concentrerons sur quelques unes dans cette rapport. Ces modèles présentent l'évalutaion des modèles de réseaux neuronaux dans le domaine de la recherche sur le QA durant ces dernières années.

3.2.1 L'ensemble de réseaux de mémoire

Les réseaux de mémoire(MemNN) stockent toute la séquence d'entrée en mémoire et exécutent une softmax fonction sur les états cachés pour mettre à jour le contrôleur (Sainbayar S., Jason W. etc, 2015).

(Weston et al., 2015) a conçu trois modèles pour tester le cardre d'evaluation, les trois modèles sont basés sur le principe de "Réseaux de mémoire". Ces réseaux peuvent être compris comme un cadre de construction des modèles comprenant les cinq parties suivantes :

1. Mémoire M: la représentation de la mémoire du modèle consiste en une liste d'emplacements mémoire pouvant être lus et écrits par les composants G, O.

2. Le composant I (transformation d'entrée): transforme les phrases d'entrée en vecteurs("*feature map*") que l'espace caractéristique interne représente.

3. Le composant G (généralisation): permet de mettre à jour la mémoire M lorsque le modèle acquiert un nouveux "feature map" d'entré, c'est un stockage de mémoire.

4. Le composant O (sortie de mémoire): Selon le "feature map" de la question, la mémoire M est recherchée et le contenu de la mémoire est sorti. Il peut être compris comme l'adresse mémoire.

5. Le composant R (réponse): après avoir obtenu les "feature map" correspondant de la memoire M, il convertit dans un format spécifique, tel que du texte. Il peut être compris comme la conversion de la mémoire abstraite en représentations.

Après MemNN, on a aussi MemN2N (Sainbayar S., Jason W. etc, 2015), GMemN2N (Fei Liu and Julien Perez., 2017), DMN (Kumar, A., Irsoy, O. etc, 2016) etc. Ils suivent tous la

même structure de MemNN. MemN2N calcule d'abord la parenté des phrases dans la question et la mémoire en prenant le produit intérieur, et la phrase ayant le plus de parenté est sélectionnée comme première phrase de support pour la question donnée. La première phrase de support est ensuite ajoutée à la question et répète la même opération avec la mémoire mise à jour pour trouver la deuxième phrase de support. GMemN2N sélectionne la phrase de support de la même manière que MemN2N, mais utilise la porte pour ajouter sélectivement la question afin de contrôler l'influence de l'information de question dans la détermination de la phrase de support à l'étape suivante.

Comme la capacité de raisonnement fort dépend de la capacité du modèle à capturer séquentiellement les bonnes phrases de support qui mènent à la réponse, la chose la plus importante qui discrimine ces modèles est la manière de construire le "feature map" de sortie. à mesure que le "feature map" des entités en sortie devient plus complexe, elle peut apprendre des modèles pour des relations plus complexes. Par exemple, MemN2N, qui a les performances les plus faibles parmi les quatre modèles, mesure la relation entre la question et la phrase par le produit interne. Ci-dessous, nous aurons également des commentaires.

3.2.2 DMN

Réseaux de mémoire dynamique(DMN (Kumar, A., Irsoy, O. etc, 2016) et DMN+ (Caiming X., Stephen M. etc., 2016)) sont aussi les réseaux mémoire, ils ont fait des progrès importants.

DMN et DMN+ ajoutent un mécanisme de fragmentation de la mémoire(mémoire épisodique) et une opération de mise à jour épisodique de la mémoire qui permet au modèle d'apprendre certaines relations logiques. Ils utilisent le "feature map" de sortie en fonction de diverses relations, telles que la différence absolue, ainsi que le produit interne, pour comprendre la relation entre la phrase et la question à divers points. Beaucoup de GRU (Cho, K., van M. etc., 2014) sont utilisés dans le modèle, C'est aussi sa caractéristique. Chaque couche utilise un GRU comme codage ou décodage.

DMN et DMN+ sont deux extensions importantes du réseau de mémoire, ils ont un fort potentiel universel et peuvent gérer de nombreux problèmes dans le traitement du langage naturel, ils peuvent également être utilisé pour former des modèles articulés multi-tâches.

3.2.3 Modèle AMN

Adaptive Memory Networks (Daniel Li, Asim Kadav., 2018) est concu pour optimiser le temps d'inférence inférieurs pour les tâches sensibles aux délais d'attente. AMN traite les paires de questions d'entrée pour construire dynamiquement une architecture de réseau. Il stocke les entités avec des poids liés dans différentes banques de mémoire. En contrôlant le nombre de banques de mémoire, AMN atteint des temps d'inférence faible avec une précision raisonnable. Les méthodes du plus proche voisin ont également été explorées sur des réseaux de mémoire. Par exemple, Hierarchical Memory Networks sépare la mémoire d'entrée.

Comme les approches passées pour adresser la mémoire externe, AMN construit les nœuds d'entité de mémoire dynamiquement. Cependant, distinct des approches passées, AMN stocke les entités de l'histoire d'entrée dans le nombre variable de banques de mémoire. Les entités représentent l'état caché de chaque mot dans l'histoire tandis qu'une banque de mémoire est une collection d'entités qui sont similaires à la question. À mesure que le nombre d'entités augmente et que l'entropie au sein d'une même banque devient trop élevée, AMN apprend à construire de nouvelles banques de mémoire et à copier des entités plus pertinentes pour une seule banque. Par conséquent, en limitant l'étape de décodage à un nombre dynamique de banques de mémoire pertinentes, AMN atteint des temps d'inférence inférieurs.

AMN est un modèle formé de bout en bout avec des paramètres appris dynamiquement pour la création d'une banque de mémoire et le mouvement des entités.

3.2.4 Modèle RMN

Relation Réseau de mémoire (RMN) (Jihyung Moon, Hyochang Yang et Sungzoon Cho., 2018) s'appuie sur le principe de réseau de relation(RN) (Adam S., David R. etc., 2017) et intègre le réseau de relation et le réseau de mémoire.

Le réseau de relation a été proposé comme solution générale au raisonnement relationnel. La philosophie de conception sous-jacente est de capturer directement la relation de soutien entre les phrases à travers le perceptron multi-

couche (MLP). Malgré sa simplicité, RN atteint de meilleures performances que les modèles précédents sans défaillance catastrophique.

Les auteurs de RMN ont trouvé que RN peut également être interprété en termes de MemNN. Mais ils ont aussi découvert que quand on utilise le RN d'origine pour établir directement des relations de phrases, s'il y a trop de phrases, ce sera un désastre. Donc ils ont integré le réseau de mémoire, RMN est capable de trouver une relation complexe, même lorsque beaucoup d'informations sont données. RMN utilise MLP pour trouver des informations pertinentes avec une nouvelle généralisation qui efface simplement les informations déjà utilisées. En d'autres termes, la RMN hérite de le "feature map" de sortie basée sur MLP de RN sur l'architecture de réseau de mémoire. le RMN a atteint l'état d'art cet année.

3.2.5 Evaluation

Nous avons collecté les résultats des testes de (Jihyung Moon, Hyochang Yang et Sungzoon Cho., 2018) et (Daniel Li, Asim Kadav., 2018) afin de comparer tout les modèles qu'on vient de citer dont le tableau 3.2.5 montre la moyenne de pourcentage d'erreur pour chaque système et le nombre de tranches echouées (une tranche est echouée si le taux d'erreur dépasse 5%).

Modèle	Moyenne de pourcentage d'erreur	Tranches echouées (erreur>5%)
MemNN	6.7	4
MemN2N	6.6	4
GMemN2N	3.7	3
DMN	6.4	2
DMN+	2.8	1
AMN	2.1	0
RMN	1.2	1

Table 4: Taux d'erreur pour chaque tranche de bAbI par système

3.2.6 Commentaire

Généralement, plus la tâche est difficile, plus les "*feature map*" en sortie et le composant de généralisation sont complexes pour obtenir la bonne réponse. Pour un ensemble de données expérimentant la capacité de raisonnement basée sur le texte du modèle, la précision globale pourrait être augmentée dans l'ordre de MemN2N, GMemN2N, DMN, DMN + et RMN, où la complexité du composant augmente. D'un autre point de vue, AMN réduit le temps d'inference et répond aux questions plus rapidement.

4 Les systèmes QA à base des graphes

Ce type des systèmes est apparut entre les systèmes pipeliné et les systèmes de l'apprentissage profond. Ils sont basés sur les relations entre les différents noeuds du graphes. Les questions posées sur ce type de systèmes sont proches des questions posées sur les modèles de l'apprentissage profond puisque ce système a comme *input* des passages de textes et une questions en langages naturel. Afin de pouvoir répondre aux questions un système QA basé sur les graphes doit tout d'abord traiter les textes en input en extrayant les relations entre les différentes entités du texte. Une relation est représentée sour la forme <r,ent1,ent2> où r est la relation et ent1 et ent2 sont les entités de cette relation. (Venna et al., 2017) ont proposé l'algorithme ci-dessous pour établir ce système.

Input : Tranche de texte & Question en langage naturelle
Output : Réponse sur la question
Étape 1: Traitement du texte et extraction des triplets.
Étape 2: Generer un multigraph G pour le texte en utilisant les triplets.
Étape 3: Analyser la question et extraire le triplet de la requete r.
Étape 4:
pour chaque sous-graphe g **dans** G **faire**
  Si g répond à q **alors**
    identifier le triplet qui répond à la question pour l'envoyer en output
  Sinon si il existe une variante t du verbe de q dans un autre temps dans un nœud dans G **alors**
    Remplacer le verbe dans q avec t et revenir à l'étape 4
  Sinon si il existe un synonyme s du verbe de q dans un nœud de G **alors**
    Remplacer le verbe par le synonyme et revenir à l'étape 4
  Sinon
    Aucune réponse
  Fin si
Fin pour

(Venna et al., 2017) ont testé leur système avec

60 document et 300 questions, avec seulement 200 questions qui ont des réponses dans les textes, en input. Ils ont aussi appliqué ces testes sur des systémes à base des règles (QUARC) et des systèmes de machine learning (AQUAREAS) pour calculer le taux d'exactitude pour chaque système c'est à dire :

$$\frac{TP+TN}{T}$$

avec :

TP : le nombre de questions correctement répondus

TN : le nombre de questions qui n'ont pas de réponse sur la somme

T : le nombre total des questions.

Le système basé sur les graphes a obtenu les meilleurs résultat comme le montre le tableau 5.

Table 5: Résultat des testes faites par (Venna et al., 2017)

Bases du système	Graphe	Règles	Machine learning
Pourcentage de précision	79.67%	40%	39.3%

5 Conclusion

Dans ce rapport, nous avons tout d'abord présenté les systèmes de QA basé sur le NLP et les types des questions qu'on peut poser sur ce genre de systèmes. Ensuite, nous avons présenté les systèmes basés sur l'apprentissage profond et quelques tranches de textes et de questions qui ont été utilisées pour évaluer ces systèmes et les résultats de cette évaluation. Finalement, nous avons présenté le principe des systèmes QA basés sur les graphes et les types des questions qu'on peut poser sur ces systèmes et les résultats de l'évalution de ces systèmes.

References

Magnini, Bernardo, Danilo Giampiccolo, Pamela Forner, Christelle Ayache, Petya Osenova, Anselmo Pe nas, Valentin Jijkoun, Bogdan Sacaleanu, Paulo Rocha, and Richard Sutcliffe 2006. *Overview of the clef 2006 multilingual question answering track. In Proceedings of the Cross-Language Evaluation Forum 2006 workshop (CLEF)*

Yutaka Sasaki, Hsin-Hsi Chen, Kuang-Hua Chen and Chuan-Jie Lin 2005. *Overview of the ntcir-5 cross-lingual question answering task (clqa1). In Proceedings of the 5th NTCIR Workshop Meeting (NTCIR-5)*

Ellen M. Voorhees. 2000. *Overview of the TREC-9 Question Answering Track*

David Elworthy. 2001. *Question answring using a large NLP system*

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y.. 2014. *On the properties of neural machine translation: Encoder-decoder approaches.*

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. *End-to-end memory networks.*

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. 2015. *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*

Goodwin, Travis R and Harabagiu, Sanda M. 2016. *Medical question answering for clinical decision support.*

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. 2016. *Ask me anything: Dynamic memory networks for natural language processing*

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. *Dynamic memory networks for visual and textual question answering.*

Fei Liu and Julien Perez. 2017. *Gated end-to-end memory networks.*

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. *A simple neural network module for relational reasoning.*

Veena G., Deepa Gupta, Athulya S., Salma Shaji 2017. *A graph-based relation extraction method for question answering system*

Jihyung Moon, Hyochang Yang, Sungzoon Cho. 2018. *Finding remo(related memory object): A simple neural architecture for text based reasoning*

Daniel Li, Asim Kadav. 2018. *Adaptive Memory Networks*