Agents conversationnels vocaux à base de réseaux de neurones artificiels profonds: un survol

Guillaume Chevalier
Université Laval
Baccalauréat en génie logiciel
guillaume.chevalier.2@ulaval.ca

Samuel Cabral Cruz
Université Laval
Baccalauréat en génie logiciel
samuel.cabral-cruz.1@ulaval.ca

Abstract

Les approches par réseaux de neurones ont récemment surpassées les approches par algorithmes classiques pour la majorité des problèmes du traitement de la langue naturelle lorsqu'assez de données sont disponibles. C'est principalement ce qui explique pourquoi nous voyons de plus en plus de solutions commerciales implémentant des agents conversationnels, tels que **Siri (Apple)** [1], **Alexa (Amazon)** [2] et **Google Assistant(Google)** [3]. Alors, comment créer son propre agent conversationnel à partir de publications récentes et de technologies de pointe ? Ainsi, nous regroupons et expliquons tous les sous-systèmes nécessaires à la construction d'un tel agent qui sera en mesure de recueillir une commande vocale pour ensuite renvoyer une réponse provenant d'une recherche dans une base de données de connaissances en texte naturel, telle que **Wikipédia**. Un aspect intéressant de la solution proposée est que la majorité des méthodes employées se basent sur les réseaux de neurones artificiels, regroupant ainsi une base de connaissances commune à chacune des étapes du traitement facilitant par le fait même l'intégration et le maintien d'une telle architecture. Dans des travaux ultérieurs, une telle architecture pourrait éventuellement être intégrée en un seul gros réseau de neurones profonds bout à bout plutôt qu'en plusieurs réseaux distincts juxtaposés les uns aux autres.

1 Introduction

Depuis la naissance de l'informatique, l'humain a toujours convoité l'idée de pouvoir interagir verbalement avec un ordinateur, et ce, de manière transparente comme s'il s'agissait d'un autre humain apte à capter la majorité des nuances du discours entretenu. Les premières tentatives (**ELIZA** - *the artificially-intelligent psychiatrist* Weizenbaum (1966) et **PARRY** - *the paranoid computer*1 Cerf (1973)) ont cependant démontré que cette tâche était loin d'être simple et qu'aucun algorithme ne pourra éventuellement répondre parfaitement à cette tâche et ainsi satisfaire les exigences du test de Turing (1950). Bien que ce sujet aura fait couler beaucoup d'encre et fait tourner les têtes, il n'existe toujours pas, à ce jour, une formule secrète parfaite pour parvenir à sa réalisation. Au cours des années, différentes démarches ont été proposées. Traditionnellement, des approches algorithmiques étaient favorisées et certains projets se fondent encore sur ces dernières, tel que **Watson (IBM)** [4] Ferrucci (2011). Ces approches ont toutefois le défaut d'être longues et ardues à développer. De plus, la réutilisation des travaux sous-jacents est plus complexe en raison du caractère sur-mesure de la solution étroitement liée au champ d'application spécifique, tel que de jouer à **Jeopardy** [5]. Depuis 2014, ce approches classiques furent appelées à être remplacées par des approches fonctionnant par réseaux de neurones profonds, le point marquant de ce virage étant la découverte des mécanismes d'attention par Bahdanau et al. (2014).

De ce fait, des approches mettant en jeu des réseaux de neurones artificiels ont fait leur apparition et ont immédiatement connu beaucoup de succès. Une des premières démonstrations des capacités de ces approches se firent sur la traduction automatisée Bahdanau et al. (2014).

1. https://www.apple.com/ca/ios/siri/
2. https://developer.amazon.com/alexa
3. https://assistant.google.com/
4. https://www.ibm.com/watson/
5. https://www.jeopardy.com/

De tels systèmes sont désormais utilisés chez **Google** pour la mise en production du fameux **Google Translate** Wu et al. (2016). Par surcroît, cette même compagnie utilise aussi des solutions neurales de *Text-To-Speech* (TTS) telles que celle proposée par Chan and Lane (2015) afin de pouvoir générer des sous-titres automatiquement pour des médias numériques [6] et analyser ces derniers afin de les regrouper sous la forme d'un arbre sémantique. Il ne s'agit ici que de simples morceaux d'un puzzle entier, soit la création d'un agent conversationnel basé sur des réseaux de neurones. C'est justement l'assemblage de découvertes provenant de différents partis (tous ayant des objectifs distincts) qui rend la conception d'un tel agent aussi complexe et l'une des raisons pour laquelle le mouvement *open-source* est si proéminent dans ce domaine.

Dans le cadre d'un échange verbal entre deux êtres, une multitude de tâches sont accomplies sans même en être conscient. Le tout débute lors d'un contact initial le plus souvent dans une forme sonore vers un destinataire. En partant de ce point, il faut premièrement capter le message, l'interpréter en mots et, malgré des obstacles environnants et culturels variés réduisant la qualité de cette transmission, filtrer ce qui est réellement important dans le signal tout en le décodant selon un dialecte particulier. C'est à cette étape que s'insère les architectures de *Speech-To-Text* (STT). Une fois en possession de ce message, il faut établir des liens entre l'énoncé qui a été donné et un registre de connaissances en plus de prendre en compte les discussions passées avec le même interlocuteur. C'est alors qu'il est possible d'établir la réponse la plus appropriée compte tenu d'une panoplie de facteurs comme l'identité de notre interlocuteur, son domaine de travail, son niveau d'éducation et même les valeurs qui sont partagées ou distinctes entre les deux partis. Cette phase implique des réseaux de neurones capables d'analyser du texte à partir d'une requête, tels que les systèmes basés sur des améliorations et une exploration des mécanismes d'attention Luong et al. (2015) ensuite appliquées à cette nouvelle tâche, introduits dans les travaux de Hermann et al. (2015). L'attention étant maintenant bien attribuée dans le texte, il sera alors possible de générer une réponse de manière similaire à ce qui est présenté dans les recherches de Serban et al. (2016) et de Serban et al. (2017).

Une fois cette réponse textuelle en main, il ne reste plus qu'à la convertir en audio, ce qui est dorénavant possible en temps réel avec une approche par réseaux de neurones convolutionnels. Un des outils les plus connus à cet effet est `Wavenet` par van den Oord et al. (2016). [7] D'autre part, ce genre de systèmes est suffisamment flexible pour opérer en plusieurs langues si il est combiné à un module de traduction automatisée. Au final, cela demande beaucoup de données d'apprentissage. Pour rajouter encore plus de difficulté, ces étapes sont à faire dans un intervalle de temps très court. Heureusement, l'étape la plus longue est de faire apprendre aux réseaux de neurones ce qu'ils ont à apprendre, tâche pouvant être réalisée à priori et réutiliser à répitition une fois complétée. Ils seront ensuite très performants lors de l'étape d'inférence, où ils analyseront et génèreront réellement de l'information en production.

2 Développement

Toutes les parties nécessaires pour concevoir un agent conversationnel basé sur une architecture neurale existent présentement. Certaines techniques se sont démarquées au fil des études. Un survol succinct des techniques les plus prometteuses est fait, de façon à ce qu'il soit possible de joindre ces dernières ensemble ce qui permettrait d'en faire une implémentation réelle et complète. Les approches neurales les plus aux goûts du jour sont à favoriser et sont introduites dans cet article. Ces approches imitent la nature et correspondent, à ce jour, à la forme la plus répandue d'intelligence artificielle. Il ne reste qu'à les informatiser convenablement et à découvrir les bonnes configurations neurales dans le but de créer l'interface conversationnelle idéale.

2.1 Traitement d'un intrant vocal

La première étape de calcul au sein d'une architecture neurale destinée à comprendre et répondre à un utilisateur est justement de comprendre

6. Comme **YouTube** le fait sur l'ensemble des vidéos pour suggérer du contenu à ses utilisateurs

7. Encore une fois développé chez **Google** pour faire du TTS, l'inverse du STT

ce qu'il dit. Pour accomplir cette tâche, il est possible d'utiliser un réseaux de neurones *Time Convolution* (TC)-*Deep Neural Network* (DNN)-*Bidirectional Long Short-Term Memory* (BiLSTM)-DNN, c'est-à-dire des convolutions temporelles (TC) suivies de couches de neurones linéaires profondes (DNN), d'un *Long Short-Term Memory* (LSTM) Bidirectionnel (BiLSTM) et puis d'un second DNN Chan and Lane (2015). Ainsi, cette architecture dépend d'un pré-traitement du signal par un autre algorithme lequel est plus classique et permet de transformer le signal en un domaine de fréquences personnalisées. C'est ce prétraitement de l'information qui est introduit dans le réseau de neurones profonds, afin d'en analyser le sens et de le pouvoir convertir en états acoustiques, lesquels peuvent être convertis, cette fois, en texte littéraire. Cette architecture neurale, imagée à la Figure 1, obtient un *Word Error Rate* (WER) de retranscription de 3.47, ce qui est présentement le *State-Of-The-Art* (SOTA) sur le jeu de données du problème du *Wall Street Journal* (WSJ) eval'92.

2.2 Extraction des composantes de l'intrant et des sources d'information à analyser

Une fois que la requête de l'utilisateur est convertie sous une forme textuelle facilement manipulable par un ordinateur, il est possible, dès lors, d'utiliser un *embedding* induit par l'étape précédente. Une autre approche consiste à reprendre cette sortie pour ensuite la fournir à une nouvelle structure qui se chargera d'aller extraire de nouvelles composantes qui aideront certainement à obtenir de meilleurs résultats pour la suite du processus.

À ce stade, nous devons comprendre que le signal est encore purement textuel et nous n'avons pour seule information qu'une décomposition des mots qui forment la demande reçue. Cependant, les langages sont formés de davantage de subtilités qu'un simple enchaînement de mots les uns après les autres. En effet, chaque mot joue un rôle précis dans la structure de la phrase et apporte une nuance particulière au contexte générale de celle-ci ou encore du texte avec une plus faible portée. C'est exactement ce que les travaux de Mikolov et al. (2013) visaient à faire. Ainsi, en 2013, ce groupe de chercheurs a fait la publication d'un article détaillant leur approche en comparant plusieurs modèles comprenant autant des approches classiques que des approches neurales. En plus de faire état de leurs travaux, ce groupe est aussi à l'origine d'outil qui est encore à ce jour considéré comme un incontournable : `word2vec`.

Malgré le fait que cet article porte sur les approches neurales, cet outil a plutôt prouvé que des modèles plus simplistes et classiques sont parfois plus appropriés. En ce qui concerne `word2vec`, ce dernier se fonde sur la combinaison de deux techniques nommées *Continuous Bag Of Words* (CBOW) (Figure 2) et Skip-gram (Figure 3). Alors que le Skip-gram se concentre à essayer de prédire le contexte environnant d'un mot donné, le CBOW cherchera plutôt à prédire la valeur considérée à partir de son environnement accordant ainsi plus d'importance à la structure des phrases plutôt qu'au contexte de son utilisation.

En fournissant la requête reçue à cet outil, il est donc possible d'extraire les composantes sémantiques et syntaxiques sous-entendues dans cette dernière. Par la suite, ces nouvelles composantes seront combinées à celle déjà obtenues à l'étape précédente. En procédant avec cette seconde approche, un gain majeur est réalisé au niveau de la performance des étapes de modélisation subséquentes en raison de l'ajout de dimensionnalités qui fourniront beaucoup plus de flexibilité pour détecter les nuances du langage. À titre d'exemple, lorsqu'un utilisateur demandera à l'assistant si ce dernier peut lui indiquer l'horaire du cinéma le plus prêt de sa position, l'assistant devra comprendre la nuance que ce qui intéresse vraiment l'utilisateur est l'horaire et non pas l'évaluation booléenne de sa capacité à s'acquitter de cette tâche. Par contre, dans le cas où l'utilisateur demanderait à l'assistant si ce dernier peut le connecté à l'Internet, l'assistant devra dans ce cas faire l'évaluation de sa capacité et répondre par affirmation à ce cher utilisateur.

Mais qu'en est-il de nos sources d'informations ? En fait, le processus entier bénéficiera qu'un travail similaire soit réalisé à ce niveau aussi. Pour ce faire, deux alternatives s'offrent encore à nous. La première consistant encore une fois à utiliser `word2vec` et la seconde repose sur le même principe, mais à un niveau supérieur

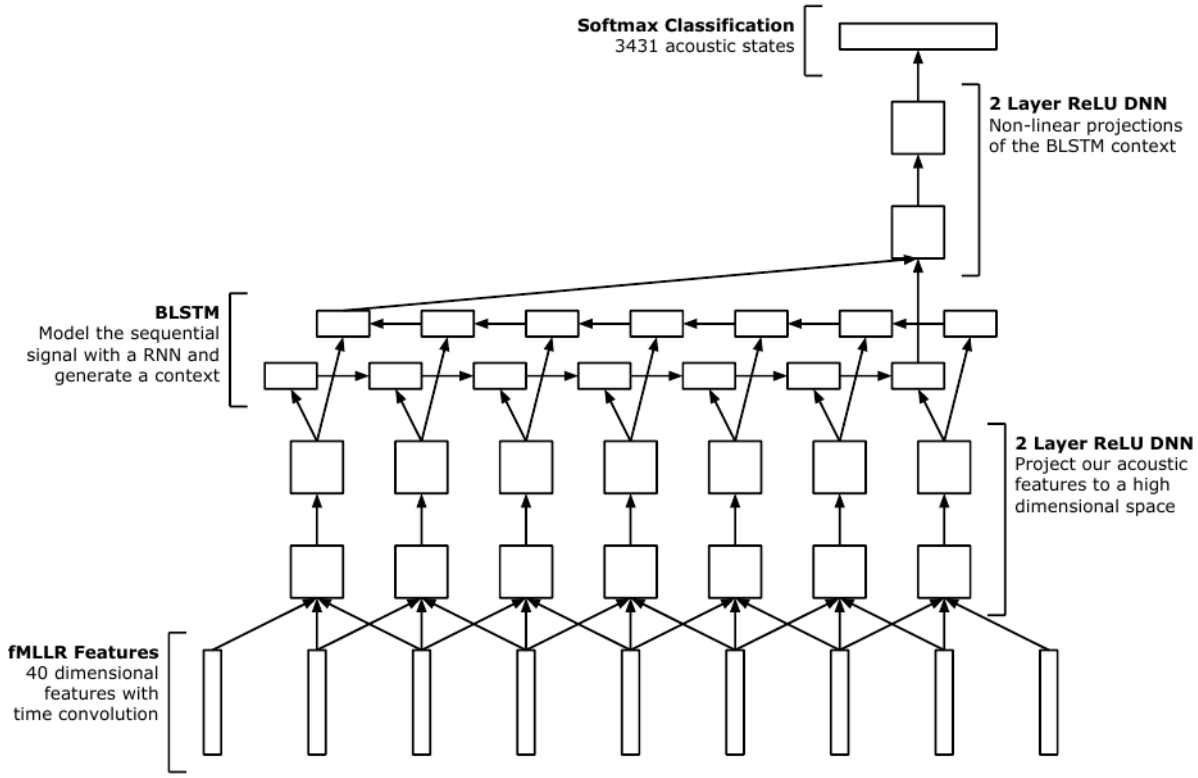

FIGURE 1 – L'architecture neurale TC-DNN-BiLSTM-DNN permet d'écouter le signal sonore à l'aide des données audio extraites en fMLLR. Ainsi, un DNN suivi d'un BiLSTM peut analyser ce signal pour classifier le tout en états acoustiques, lesquels sont eux-mêmes repris par un algorithme classique qui permet de rassembler ces états en mots réels. Notons que cette architecture neurale peut être utilisée pour raffiner le signal des mots prononcés, ce qui peut être envoyé directement dans un réseaux de neurones supérieur en tant que *embedding*. [3]

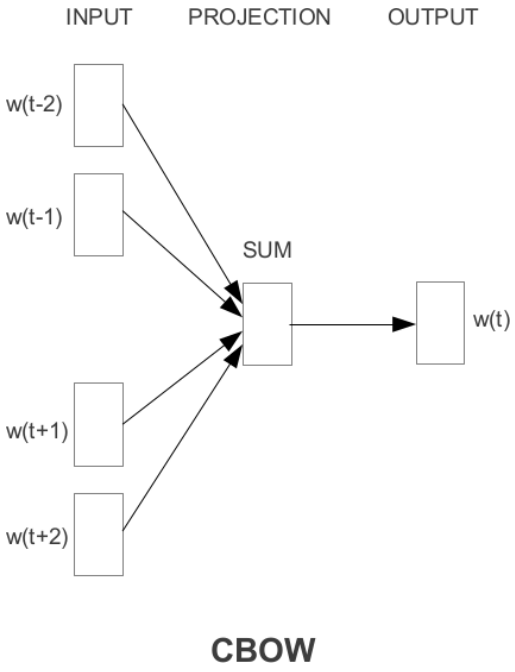

FIGURE 2 – Architecture de la méthode de prédiction CBOW [10]

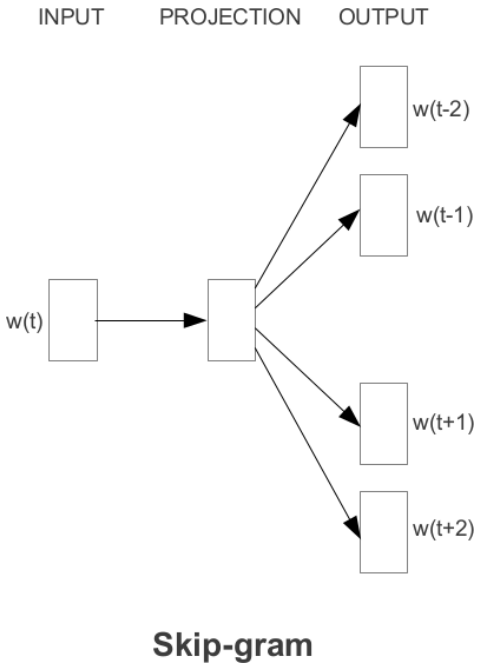

FIGURE 3 – Architecture de la méthode de prédiction Skip-gram [10]

d'abstraction en considérant cette fois l'utilité de chacune des phrases dans le texte plutôt que de se concentrer sur le rôle de chaque mot dans chaque phrase Conneau et al. (2017). De plus, il est possible d'utiliser les représentations neurales intermédiaires du réseaux de neurones du système précédent dans le flût d'information afin d'y extraire des informations sur la personne qui parle provenant du contexte vocal plutôt que textuel.

En somme, toutes ces composantes ainsi dérivées pourront ensuite être fournies en entrée d'un réseau de neurones tel qu'un *Recurrent Neural Network* (RNN) comme il est expliqué à section suivante. En effet, un RNN peut lire des mots une fois les mots transformés en *embedding*, ce qui est propice à une utilisation neurale de ces mots.

2.3 Interpréter la requête et cibler le contenu d'intérêt pour y répondre

Pour analyser les demandes de l'utilisateur, il faut les traduire en requêtes neurales pour ensuite rechercher dans le texte les passages intéressants. Cela est une étape importante à comprendre avant d'analyser davantage ce qui sera expliqué dans les prochaines sections. C'est l'une des choses que peuvent faire les mécanismes d'attention tels qu'introduits par Bahdanau et al. (2014). Ces mécanismes sont illustrés à la Figure 4. Son fonctionnement va comme suit. En premier lieu, le texte est lu séquentiellement en entrée (en bas à gauche). Ensuite, une requête est faite pour filtrer ce qui est lu et faire l'alignement d'attention (en bas à droite). Cette requête peut provenir d'un autre réseaux de neurones, mais est ici elle-même générée dans un contexte de traduction automatique. La requête est donc de demander par quoi la phrase devrait débuter lors de ce début de traduction, ce que le décodeur (à droite) peut utiliser pour construire mot par mot une phrase traduite avec une requête à chaque nouveau mot, séquentiellement. Cette requête est à chaque étape comparée à toute l'information à filtrer par le mécanisme d'attention (au centre). C'est ainsi qu'un résultat est formulé par ce calcul du mécanisme d'attention en fonction de la requête demandée (en haut). Les auteurs soutiennent que les mécanismes d'attention sont à explorer plus en profondeur et que cela n'est que leur début. Une certaine exploration de ce mécanisme est faite par Luong et al. (2015). Dans ce papier, ils définissent, entre autres choses, un tel mécanisme comme étant un mini réseau de neurones dans un plus gros. Ce mini réseau de neurones sert à déterminer où porter de l'attention, et peut prendre des formes variées, tel qu'un réseaux de neurones linéaire à deux couches, ou bien une comparaison par produit vectoriel de chaque élément à compa-

rer à la requête, en tant que mesure de similarité entre la requête et les éléments d'information où rechercher.

Il est bien d'utiliser des mécanismes d'attention pour faire de la traduction automatique Bahdanau et al. (2014), mais il est tout aussi possible de les utiliser pour trouver dans du texte les passages intéressants en fonction d'une question ? C'est ce que font Cui et al. (2016), tout comme Xiong et al. (2016) sur le jeux de données du SQuAD Rajpurkar et al. (2016), les deux approches s'inspirant des travaux de **Google**, lesquels sont détaillés à la section suivante, Hermann et al. (2015). Les travaux de Cui et al. sont illustrés à la Figure 5.

2.4 Formulation d'une réponse à partir de l'information d'intérêt retenue

Bien qu'il est intéressant de trouver l'endroit où porter attention dans un corpus textuel, il est tout autant intéressant de savoir comment générer une réponse structurée et concise à l'utilisateur. Cela peut être fait en utilisant le *Hierarchical Recurrent Encoder-Decoder* (HRED) tel qu'introduit par Serban et al. (2016). En effet, HRED est une imbrication hiérarchique de réseaux de neurones récurrents. Un premier est utilisé afin d'encoder les phrases, un second est nécessaire afin de garder le contexte des réponses passées lesquelles ont déjà été traitées [8] et un troisième RNN est mis à profit afin de décoder l'information en une réponse à l'utilisateur. En adaptant l'architecture neurale HRED de façon à lui donner des mécanismes d'attention tels que précédemment expliqués, il est possible de générer la réponse en retour de la requête attentionnelle à l'utilisateur. Ainsi, en ayant le contexte de la question que l'utilisateur pose ainsi que le contexte des documents à parcourir avec les mécanismes d'attention, il est possible de chercher dans le texte ce qu'il faut comme information, ce qui est envoyé au décodeur du HRED lequel peut répondre avec le nouveau contexte de l'information trouvée par la recherche effectuée. Ainsi, le premier RNN du HRED qui encode l'information peut utiliser `word2vec` Mikolov et al. (2013) directement, en plus de l'*embedding* provenant de l'avant dernière couche de neurones du DNN de la partie STT. En plus de cela, il est possible d'utiliser `inferSent` de **Facebook** Conneau et al. (2017), dont la sortie pourra être concaténée au signal de sortie du RNN encodeur du HRED, en tant qu'*embedding* supplémentaire au niveau des phrases plutôt qu'au niveau des mots comme expliqué à la sous-section 2.2.

Dans une amélioration plus récente de l'architecture HRED Serban et al. (2017), présentée à la Figure 6, il est possible d'utiliser une variable latente intermédiaire laquelle permet de faire le pont entre les réponses envoyées du décodeur vers l'utilisateur, en plus de réinjécter cette réponse dans l'encodeur qui écoute la réponse de l'utilisateur après lui avoir répondu une première fois. En retournant ainsi l'information du décodeur dans l'encodeur, le contexte se retrouve à être conservé d'une phrase à la prochaine et d'ainsi avoir un discours plus fluide tout en étant moins assujettis à des variations subites de sujet ou d'interprétation. D'autre part, ce passage d'information aura pour effet de renforcer la qualité de la requête attentionnelle laquelle peut être générée à la toute fin de l'encodeur du HRED. C'est à ce moment que le mécanisme d'attention décrit dans la sous-section 2.3 portant sur l'analyse de texte suite à des questions pourrait être inséré. Une fois la question posée par l'utilisateur et lue dans l'encodeur, le HRED peut bénéficier de cette question dans son RNN intermédiaire, et ce, en tant que requête attentionnelle à passer directement au système attentionnel. Des travaux similaires ont été réalisés par Hermann et al. (2015), chez **Google**. Somme toutes, le HRED aura accès à la question de l'utilisateur et au corpus de texte dans lequel il peut maintenant cibler l'information pertinente.

Étant donné la taille énorme du corpus textuel dans lequel le réseaux de neurones peut lire l'information [9], il est possible d'appliquer un `MapReduce` Dean and Ghemawat (2008) pour ainsi améliorer les performances de ce processus et réduire de façon importante le temps de réponse de notre assistant, ce qui est un aspect primordial à son déploiement et son adoption de la part de l'utilisateur. Cette technique procède de façon distribuée sur plusieurs centaines d'ordinateurs. Dans le cas présent, ceux-ci utiliseraient eux-même les implémentations de `word2vec` et de `inferSent` sur le corpus d'information, et cela en ayant en main la requête attentionnelle générée par le mécanisme d'attention, selon les

8. Comme un suivi de la discussion dans une mémoire temporaire

9. L'ensemble du texte sur Wikipédia par exemple

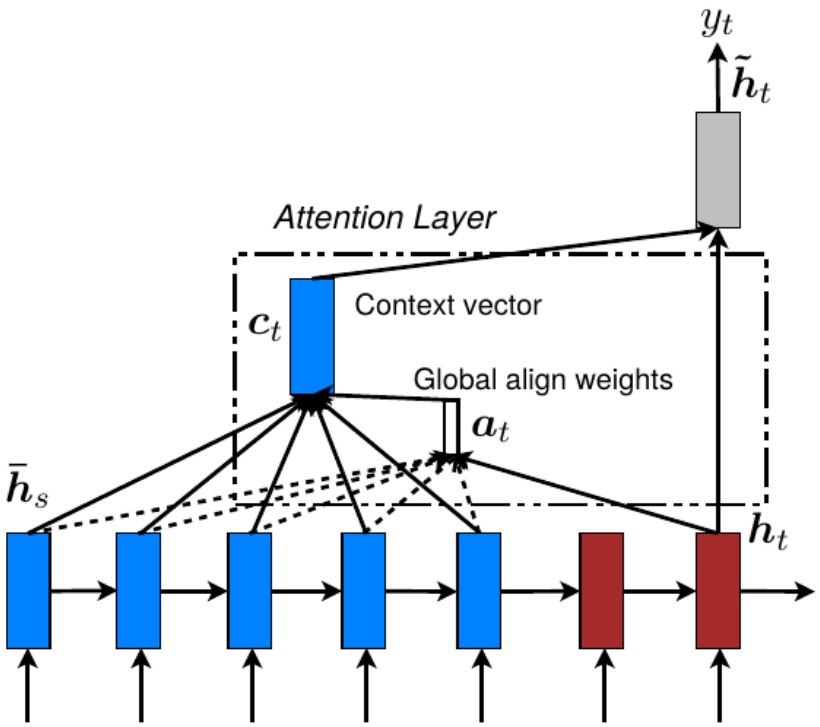

FIGURE 4 – Mécanisme d'attention sous sa forme générale, tel qu'introduit par Bahdanau et al. (2014) et ici raffinés par Luong et al. (2015) dans cette figure.

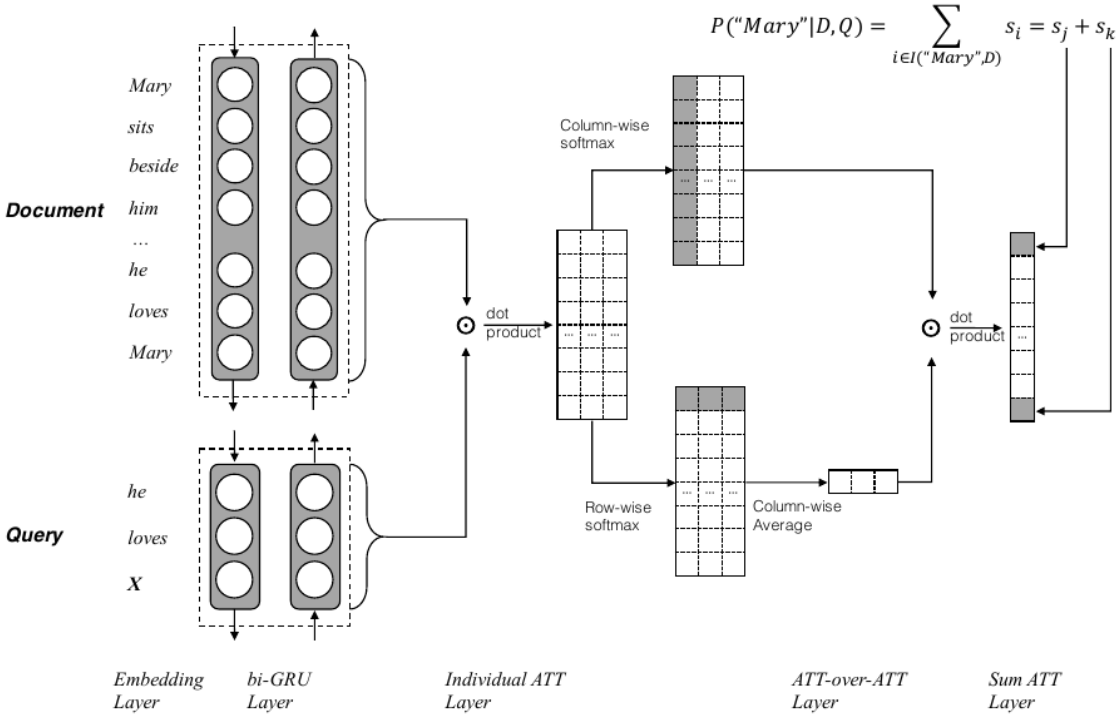

FIGURE 5 – Réseaux de neurones profonds permettant d'analyser un document (en haut à gauche) en fonction d'une requête (en bas à gauche) pour produire une réponse avec l'information trouvée (à droite) Cui et al. (2016).

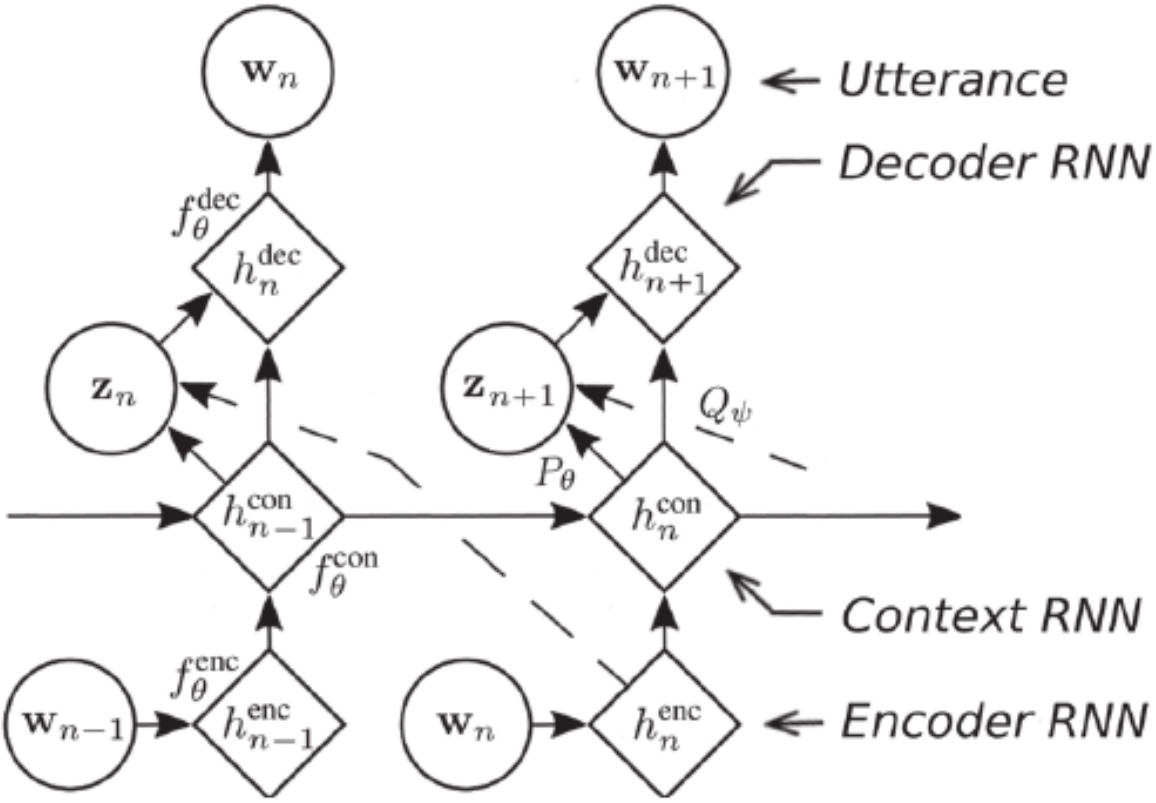

FIGURE 6 – L'architecture HRED améliorée avec une variable latente Serban et al. (2017)

principes de `MapReduce`. Cette partie, qui est distribuée et qui est surnommée, le lecteur impatient, est exposée à la Figure 7. Il existe même une amélioration possible sur cet architecture neurale. Il est visible dans la figure que plusieurs itérations entre la requête et le système attentionnel est fait. Cela devrait être simplifié en une seule étape afin de réduire la complexité algorithmique de linéaire à constante en fonction de la longueur de la requête, en termes de nombre de mots. La recherche suivant la requête pouvant être distribuée, il devient donc très rapide d'accomplir le celle-ci, tout comme l'ensemble des opérations décrites aux sections précédentes.

2.5 Retourner la réponse textuelle sous la forme d'un signal audio

Une fois une réponse générée, il ne reste plus qu'à la renvoyer à l'utilisateur sous une forme audio, ce qui est un autre calcul rapide qui peut accompli en temps réel. Cela est rendu possible avec le *Convolutional Neural Network* (CNN) `Wavenet` de van den Oord et al. (2016). En effet, il est possible de générer n'importe quel ton de voix avec `Wavenet`, option qui pourrait être laissée à la discrétion de l'utilisateur. À titre d'exemple, cette architecture neurale est tellement puissante qu'il est possible de lui faire imiter la voix du président. Cette découverte récente par **Google** marquera donc la première occurrence d'une réplique parfaite du timbre d'une voix humaine réelle plutôt qu'une voix aux allures robotiques, ainsi l'illusion est bien réussie. La façon dont `Wavenet` fonctionne est d'établir un préalable statistique (une variable conditionnée) qui est donnée à un premier algorithme qui s'occupe de trouver les bons tons de voix à générer avec `Wavenet`, à partir du texte. C'est ainsi que `Wavenet`, conditionné par le ton de voix demandé et par le texte fourni, peut générer la voix de façon réalistique. C'est une méthode point par point, où chaque point dans la vague audio est généré en fonction des points précédents et du conditionnement demandé. L'entraînement de ce réseau se fait à très bas niveau sur le signal qui est à un taux d'échantillonage de 16 kHz, ce qui est suffisant pour capturer les infimes variations de quelqu'un qui parlerait réellement dans un enregistrement. Cette phase générative est illustrée dans dans la Figure 8.

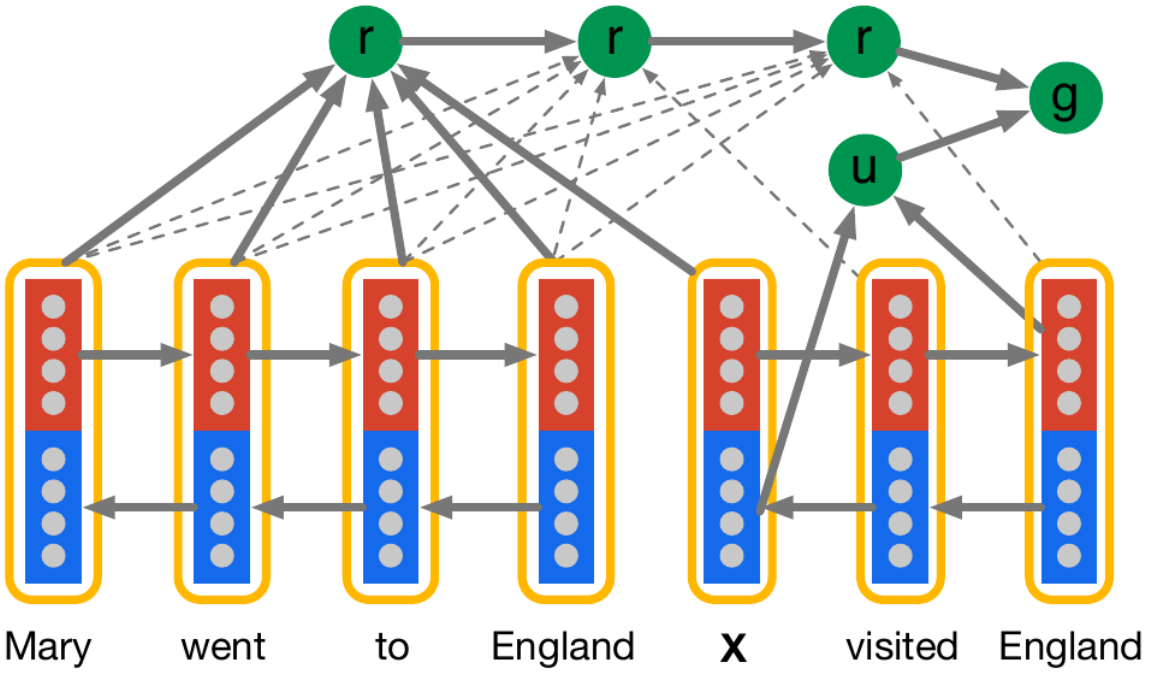

FIGURE 7 – Le lecteur impatient prends la requête *X visited England* afin de faire une recherche dans le texte *Mary went to England*, à l'aide du mécanisme d'attention lequel est ici dénoté `r`, assisté de la requête `u` Hermann et al. (2015)

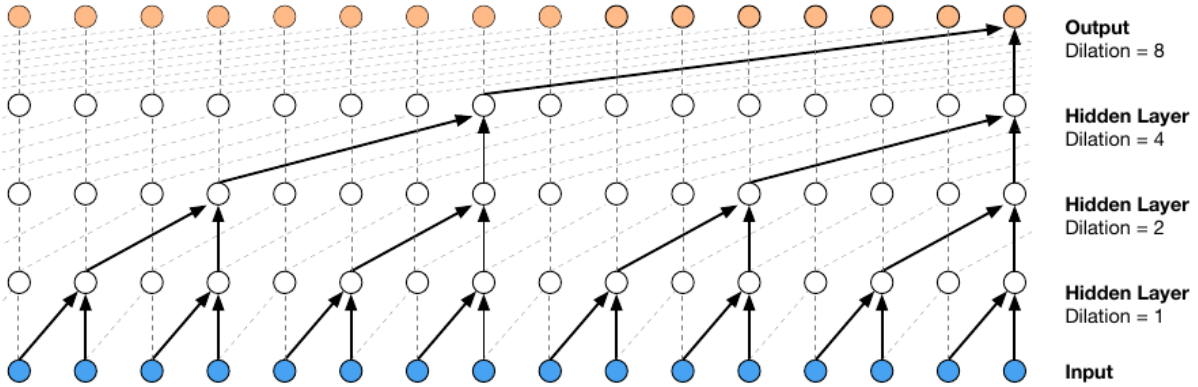

FIGURE 8 – À l'aide de convolutions causales dilatées, il est possible de prédire le prochain point dans la vague audio de façon efficace. Il s'agit d'un modèle autorégréssif : les points passés sont utilisés pour prédire les points suivants du même signal. Ainsi, la sortie est remise en entrée pour le calcul de l'étape suivante, cet échantillongage peut faire usage de mémoire cache, ce qui donne à cet algorithme un temps linéaire pour la génération dépendant de la longueur du signal à générer van den Oord et al. (2016).

Conclusion

Au terme de cet article, une revue des différents portions d'un agent conversationnel complet a été accomplie. Nous avons mentionné qu'un intrant vocal pourrait être converti sous une forme textuelle grâce à l'utilisation d'une architecture TC-DNN-BiLSTM-DNN. Il a aussi mentionné que les composantes syntaxiques et sémantiques d'un texte pouvaient être extraire grâce à `word2vec` au niveau des mots ou de manière similaire avec `inferSent` au niveau des phrases. Les méchanismes d'attentions de (CITER LES SOURCES) pourront ensuite être exploités afin d'identifier l'information qui devra être retournée à l'utilisateur. Une fois en possession de cette information, celle-ci devra être intégré dans une réponse textuelle complète respectant les règles de la langue utilisée dans l'échange. C'est à ce moment que l'architecture HRED entrera en jeu pour générer un discours fluide et cohérent. En dernier lieu, la génération d'un signal sonore artificiel sera déléguée à l'architecture `Wavenet`. Avec un peu de travail pour combiner tous ces morceaux du casse-tête, un agent conversationnel performants en résultera. En guise de conclusion, nous remarquons encore une fois que les approches par réseaux de neurones dominent encore une fois la majorité des autres approches auparavant exploitées. Ce qui a de plus extraordinaire avec ces derniers, c'est que des problèmes qui peuvent sembler immensément complexe de prime abord se révèle à être beaucoup plus simples lorsque nous laissons les machines déterminer elles-mêmes les composantes et du signal à déduire des ces dernières via des solutions semi ou non-supervisées.

Remerciements

Nous tenons à remercier Nadir Belkhiter, professeur agrégé à l'*Université Laval* (UL) et membre de l'Ordre des ingénieurs du Québec (OIQ), pour son support lors de la réalisation de cet article, et des ressources utiles lesquelles qu'il a mis à notre disposition.

Références

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Vint Cerf. 1973. PARRY encounters the DOCTOR. RFC 439.

William Chan and Ian Lane. 2015. Deep recurrent neural networks for acoustic modelling. *CoRR* abs/1504.01482.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR* abs/1705.02364.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *CoRR* abs/1607.04423.

Jeffrey Dean and Sanjay Ghemawat. 2008. Mapreduce : Simplified data processing on large clusters. *Commun. ACM* 51(1) :107–113.

David A. Ferrucci. 2011. Ibm's watson/deepqa. *SIGARCH Comput. Archit. News* 39(3).

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR* abs/1506.03340.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR* .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad : 100, 000+ questions for machine comprehension of text. *CoRR* abs/1606.05250.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th Annual AAAI Conference on Artificial Intelligence*. AAAI Press, Phoenix, Arizona USA, volume 3776 of *Special Track on Cognitive Systems*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31th*

Annual AAAI Conference on Artificial Intelligence. AAAI Press, San Francisco, California USA, volume 3295 of *Natural Language Processing and Machine Learning*.

A. M. Turing. 1950. I.—computing machinery and intelligence. *Mind* LIX(236) :433–460.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet : A generative model for raw audio. *CoRR* abs/1609.03499.

Joseph Weizenbaum. 1966. Eliza&mdash ;a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9(1) :36–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system : Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604.