Agents conversationnels à base de réseaux de neurones artificiels profonds

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 en ce qui concerne la majorité des problèmes du traitement de la langue naturelle. C'est principalement ce qui explique pourquoi nous voyons de plus en plus de solutions commerciales implémentant des agents conversationnels, tels que **Siri (Apple)**, **Alexa (Amazon)** et **Google Assistant et Google Home (Google)**.

Devons-nous toutefois croire que la réalisation d'un tel agent et une science réservée aux géants de l'industrie technologique ? Comment s'y prennent-ils ? Est-ce envisageable de se créer notre propre agent conversationnel ? Si oui, Comment ?

Dans cet article, nous tenterons de répondre à chacune de ses questions en se basant sur les parutions d'études récentes. Pour ce faire, nous déquortiquerons le problème en plusieurs sous-étapes qui devront être accomplies par notre agent conversationnel hypothétique. De cette façon, il sera beaucoup plus simple d'avoir une représentation mentale des différentes parties d'une architecture entièrement neurale pour les agents conversationnels.

1 Introduction

Depuis que la naissance de l'informatique, l'humain a toujours convoité l'idée de pouvoir intéragir verbalement avec un ordinateur, et ce, de manière totalement transparente comme s'il s'agissait d'un autre humain apte de capter la majorité des nuances du discours qu'il entretiendra. Bien que ce sujet aura fait couler beaucoup d'encre et fait tourner les têtes, nous ne connaissons toujours pas à ce jour une formule secrète 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** de **IBM**. Ces approches ont toutefois le défaut d'être longues et ardues à développer et la réutilisation des travaux sous-jacents n'est qu'encore plus complexe en raison du caractère très sur mesure du problème ou du champ d'application auquel il s'applique.

Récemment, des approches mettant en jeu des réseaux de neurones artificiels ont leur apparition et ont immédiatement connus beaucoup de succès. À titre d'exemple, en 2014, un grand pas a été réalisé est fait lorsque les techniques par réseaux de neurones viennent à dépasser les performances des techniques classiques pour la tâche de faire de la traduction automatique Bahdanau et al. (2014). C'est aussi de tels systèmes qui sont désormais utilisés chez **Google** pour la mise en production du fameux **Google Translate** Wu et al. (2016). Cette même compagnie utilise aussi des algorithmes de *Speech-to-Text* afin de pouvoir générer des sous-titres automatiquement pour les vidéos **YouTube** et afin de pouvoir analyser les vidéos et les lier entre elles avec une approche sémantique. D'ailleurs, il est dorénavant possible de générer de l'audio en temps réel avec une approche par réseaux de neurones constitutionnels van den Oord et al. (2016).

Il ne s'agit ici que de différents morceau de puzzle qui mèneront éventuellement à la création d'un agent conversationnel complet. Cet notamment ce qui explique pourquoi la création d'un tel agent est une tâche aussi compliquée. Dans

le cadre d'un échange verbal entre deux êtres, une multitude de tâches sont accomplies sans même que nous ne soyons conscient. Le tout débute lors d'un contact initial le plus souvent dans une forme auditive vers un destinataire. En partant de ce point, à titre de destinataire, nous devrons premièrement capter ce message, malgré des obstacles environnants réduisant la qualité de ce dernier, filtrer ce qui est réellement important dans le signal et le décoder selon un dialecte sous-entendu par l'emplacement sur le globe terrestre où nous nous trouvons. Une fois en possession de ce message, nous établirons des liens entre l'énoncé qui a été donné et le registre de connaissances que nous possèdons. Nous établirons ensuite quelle est la réponse la plus appropriée compte tenu d'une panoplie de facteurs comme l'identité de notre interlocuteur, nos valeurs, nos connaissances, etc.

Une fois avec cette réponse en main, nous ne sommes rendus qu'à la moitié du parcours puisque nous devrons refaire la totalité de ce trajet à l'inverse. Ainsi, nous structurerons la réponse ainsi trouvée sous une forme syntaxique et sémantique suffisamment complète afin de favoriser une compréhension immédiate en mettant à profit notre vocabulaire et les différentes règles qui définissent une utilisation adéquate du dialecte sous lequel l'échange a été initiée. Finalement, nous émettrons à notre tour ce signal vers notre interlocuteur. Bien entendu, il ne s'agit que du chemin traditionnel d'une conversation, mais il serait simple d'y intégrer les nombres processus de traductions qui sont aussi mis en jeu lorsque les deux personnes ne possède pas la même langue maternelle compliquant davantage l'ensemble du processus. Pour rajouter encore plus de difficulté, nous devrons répéter toutes ces étapes dans un interval de temps très rapide pour éviter que la conversation devienne simplement impossible à suivre ou encore que notre interlocuteur se lasse de cet échange et préfère ainsi l'interrompre.

Dans cet article, nous aborderons ainsi chacune de ses étapes en précisant comment nous pouvons parvenir à les résoudres en favorisant plus souvent qu'autrement les approches neuronales les plus aux goûts du jour.

2 Développement

Toutes les parties nécessaires pour concevoir un agent conversationnel basé sur une architecture neuronale existent présentement. Certaines techniques se sont démarquées au fil des études. Un survol rapide des techniques les plus prometteuses est fait, de façon à ce qu'il soit possible de joindre toutes ces techniques ensemble afin de créer un seul agent conversationnel complet, ce qui permettrait d'en faire une implémentation réelle complète.

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 de comprendre ce qu'il dit. Pour accomplir cette tâche, il est possible d'utiliser un réseaux de neurones TC-DNN-BLSTM-DNN, c'est-à-dire, des convolutions temporelles (TC) suivies de couches de neurones linéaires profondes (DNN), d'un LSTM Bidirectionnel (BLSTM) et puis d'un second DNN final 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 pré-traitement de l'information qui est introduit dans le réseau de neurones profond, afin d'en analyser le sens et de pouvoir convertir cela en états acoustiques, lesquels peuvent être convertis, cette fois, en texte littéraire. Cette architecture neurale, imagée à la Figure 1, obtient un WER (Word Error Rate) de retranscription de 3.47, ce qui est présentement l'état de l'art (SOTA) sur le jeux de données et problème du Wall Street Journal (WSJ) eval'92

L'architecture neurale TC-DNN-BLSTM-DNN permet d'écouter le signal audio à l'aide des données audio extraites en fMMLR. Ainsi, un DNN suivi d'un BLSTM peut analyser ce signal pour classifier cela en états acoustiques, lesquels sont eux-mêmes repris par un algorithme classique qui permet de rassembler ces états en mots réels [DeepRecurrentNeuralNetworksForAcousticModelling]. Notons que cette architecture neural 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 plongeage.

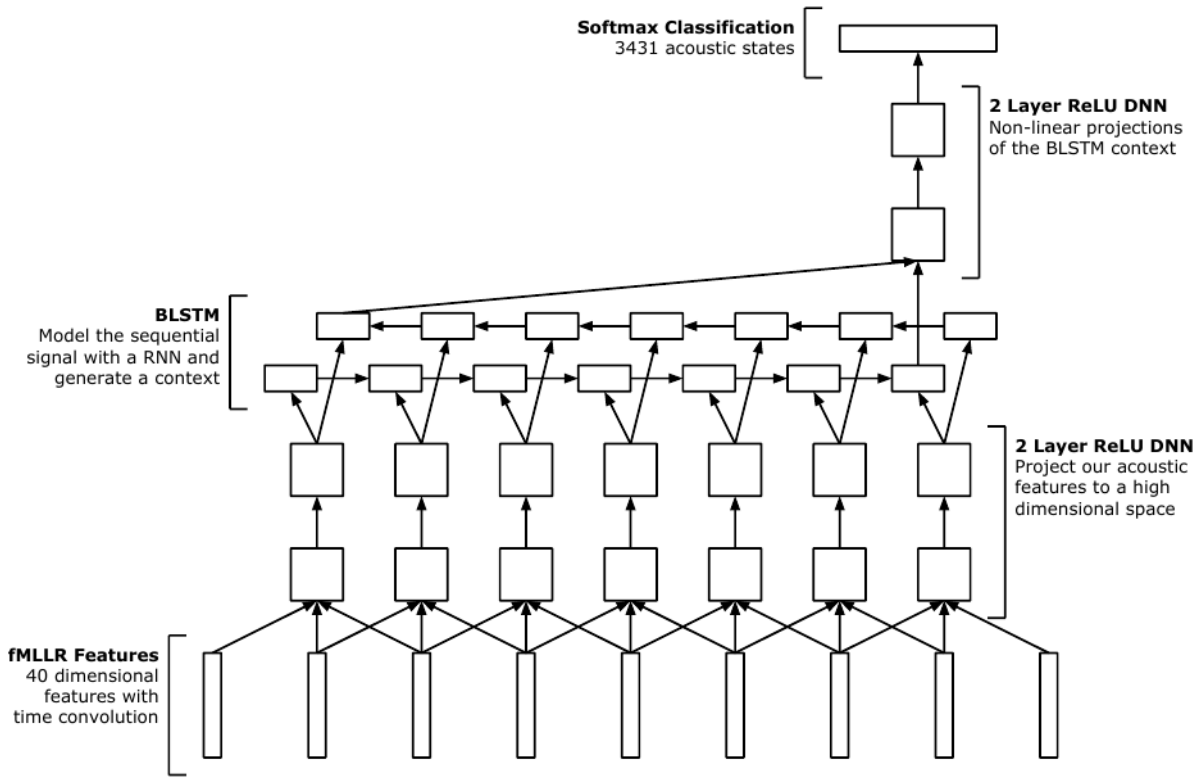

FIGURE 1 – Architecture neurale TC-DNN-BLSTM-DNN [rajouter reference ici]

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

Une fois que la requête de l'utilisateur aura été convertie sous une forme textuelle facilement manipulable par un ordinateur, nous pourrions, dès lors, utiliser le plongement 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 ... visaient à faire. Ainsi, en ..., ce groupe de chercheurs a fait la publication d'un article détaillant leur approche en comparant plusieurs modèles différents comprenant autant des approches classiques que des approches neuronales. 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 : Mikolov et al. (2013).

Malgré le fait que cet article porte sur les approches neuronales, cet outil a plutôt prouvé que des approches plus simplistes et classiques sont parfois plus appropriées. Word2vec se fonde sur la combinaison de deux approches nommées *Continuous Bag Of Words* (Figure 2) et *Skip-gram* (Figure 3). Alors que le *Skip-gram* se concentre à essayer de prédire son contexte, 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 d'utilisation.

En fournissant la requête reçue à cet outil, il sera donc possible d'extraire les composantes sémantiques et syntaxiques sous-entendues par cette dernière. Par la suite, ces nouvelles composantes seront combinées à celle que nous avions déjà obtenues à l'étape précédente. En

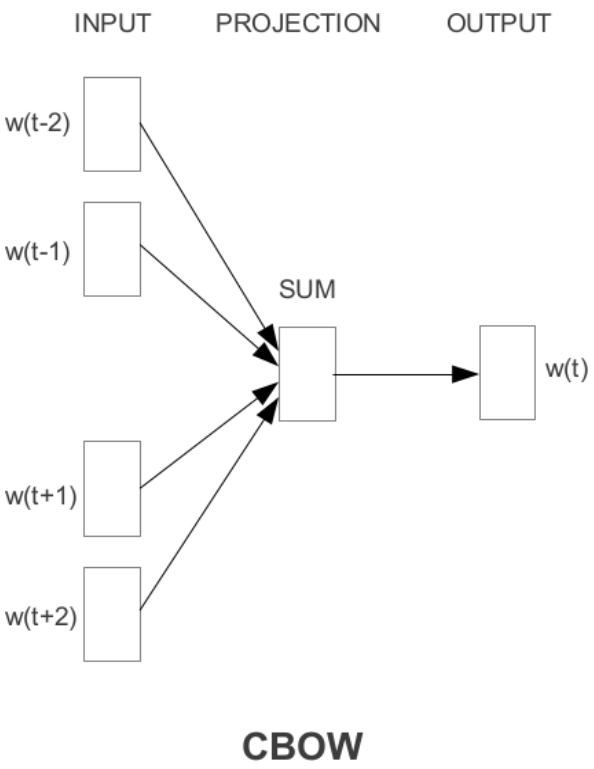

FIGURE 2 – Architecture de la méthode de prédiction CBOW [5]

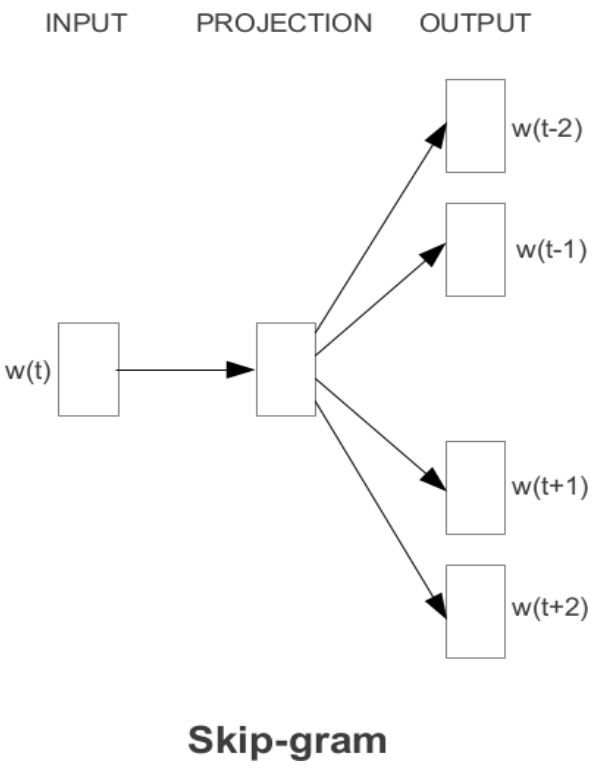

FIGURE 3 – Architecture de la méthode de prédiction Skip-gram [5]

procédant avec cette seconde approche, nous réaliserons certainement un gain majeur au niveau de la performance des prochaines étapes en raison de l'ajout important de dimensionnalités qui fourniront beaucoup plus de flexibilité aux réseaux de neurones suivantes qui devront à leur tour 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 en par affirmation à notre cher utilisateur.

Mais qu'en est-il de nos sources d'informations ? En fait, le processus entier bénéficiera certainement qu'un travail similaire soit fait à ce niveau aussi. Pour ce faire, deux approches 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 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).

En somme, toutes ces composantes ainsi dérivée pourrons ensuite être fournies en entrée d'un réseau de neurones tel qu'un RNN tel qu'il sera expliqué à section suivante.

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

1. Mécanismes d'attention : expliquer et introduire les papers. Faire ca dans l'intro ? ? 2. QA pur et dur

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

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 Iulian V. Serban et al. 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 traitées, comme un suivi de la discussion dans une mémoire temporaire, et finalement un troisième RNN est mis à profit afin de décoder l'information en une réponse à l'utilisateur. En insérant une telle architecture neurale dans les concepts précédemment abordés, il est possible de générer la réponse en retour à l'utilisateur en ayant le contexte de la question qu'il pose ainsi que le contexte des documents à parcourir avec les mécanismes d'attention, tel que décrit dans la section précédente sur la pose de question. Ainsi, le premier RNN du HRED qui encode l'information peut utiliser word2vec Mikolov et al. (2013) directement, en plus d'utiliser un *embedding* provenant de l'avant dernière couche de neurones du DNN (Deep Neural Network) de STT (Speech to Text). En plus de cela, il est possible d'utiliser le réseau de neurones infersent de **Facebook** Conneau et al. (2017), lequel peut être concaténé au signal de sortie du RNN encodeur du HRED, en tant que plongement supplémentaire au niveau des phrases plutôt qu'au niveau des mots.

Dans une amélioration plus récente de l'architecture HRED Serban et al. (2017), 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 suite à cela. En retournant ainsi l'information du du décodeur dans l'encodeur, nous nous assurons de conserver le contexte 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 section précédente 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 Karl Moritz Hermann et al. chez **Google** Hermann et al. (2015). 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, tel que l'ensemble du texte sur Wikipédia par exemple, il est possible d'appliquer un MapReduce pour ainsi améliorer les performances de ce processus et ainsi de façon important le temps de réponse de notre assistant ce qui est un aspect primordial. Cette technique procède de façon distribuée sur plusieurs centaines d'ordinateurs lesquels utilisent eux-même les implémentations de word2vec et infersent sur le corpus d'information ainsi que la requête attentionnelle en tant que préalable. Cette partie, qui est distribuée et qui est surnommée, le lecteur impatient, est représenté à la Figure 4. Il y a 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 fait 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 suite à la requête pouvant être distribuée, cela peut être fait en un temps très rapide, tout comme l'ensemble des opérations décrites dans les sections précédentes.

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

Une fois une réponse générée, il est intéressant de générer l'audio de cette à nouveau afin de répondre à l'utilisateur, ce qui est un autre calcul rapide qui peut se faire en temps réel. Cela est possible avec le CNN (Conv.. neural net) Wavenet d'Aaron van den Oord et al., développé chez Google van den Oord et al. (2016). En effet, il est possible de générer n'importe quel ton de voix avec Wavenet, ainsi le choix de la voix de la personne qui parle peut être fait par 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 est la première fois qu'il est possible de confondre la voix pour une voix humaine réelle plutôt qu'une voix robotique, 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

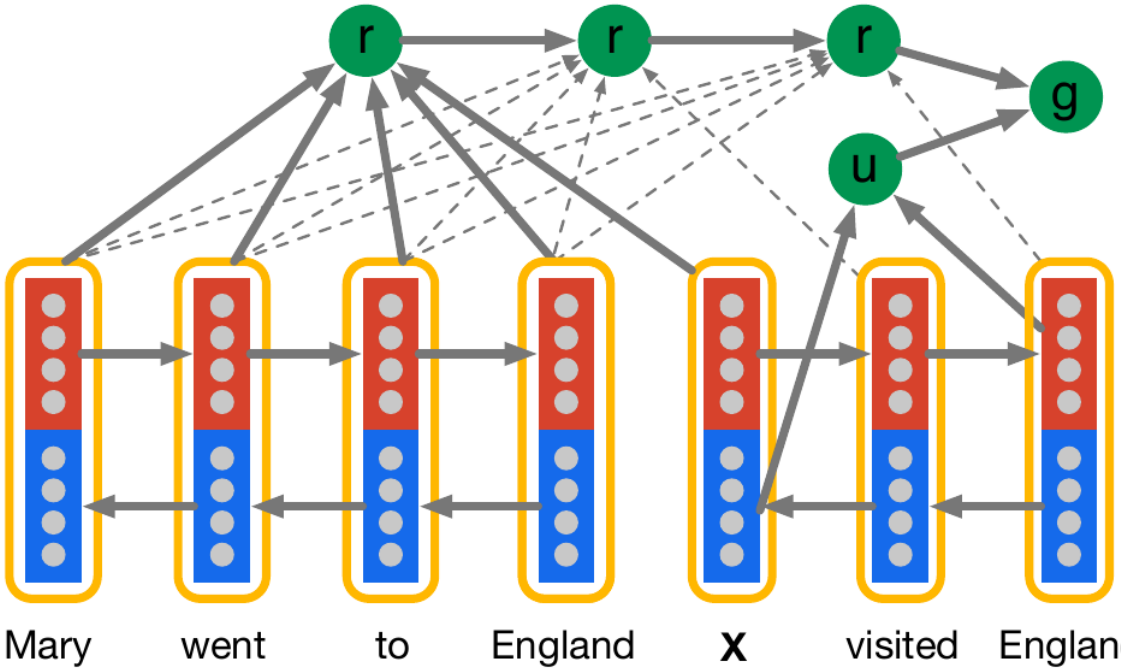

FIGURE 4 – Le lecteur impatient prends la requête “X visited England” afin de faire une recherche dans le texte “Mary went to England”, à l’aide du mécanisme d’attention lequel est ici dénoté “r”, assisté de la requête “u” [rajouter reference ici]

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 lui-même par le ton de voix demandé et par le texte, peut générer la voix de façon réalistique. C’est une méthode point par point, ainsi, chaque point dans la vague audio est généré en fonction des points précédents et du conditionnement demandé, c’est très bas niveau sur le signal qui est à un taux d’échantillonage de 16 kHz lors de l’entraînement, ce qui est assez pour capturer les subtilités de quelqu’un qui parlerait réellement dans un enregistrement. Cette phase générative est illustrée dans dans la Figure WAVENET.

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 *Time Convolution* (TC)-*Deep Neural Network* (DNN)-*Bidirectional Long Short-Term Memory* (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 et membre de l'Ordre des ingénieurs du Québec (OIQ), pour son support lors de la réalisation de cet article.

Références

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

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.

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.

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

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*.

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.

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.