Apprentissage en ligne interactif d'un générateur en langage naturel neuronal pour le dialogue homme-machine

Matthieu Riou Bassam Jabaian Stéphane Huet Fabrice Lefèvre
CERI-LIA, Université d'Avignon, France
matthieu.riou@alumni.univ-avignon.fr, {bassam.jabaian, stephane.huet, fabrice.lefevre}@univ-avignon.fr

RÉSUMÉ

Récemment, de nouveaux modèles à base de réseaux de neurones récurrents ont été proposés pour traiter la génération en langage naturel dans des systèmes de dialogue (Wen *et al.*, 2016a). Ces modèles demandent une grande quantité de données d'apprentissage ; or la collecte et l'annotation de ces données peuvent être laborieuses. Pour répondre à cette problématique, nous nous intéressons ici à la mise en place d'un protocole d'apprentissage en ligne basé sur un apprentissage par renforcement, permettant d'améliorer l'utilisation d'un modèle initial appris sur un corpus plus restreint généré par patrons. Dans cette étude exploratoire, nous proposons une approche basée sur un algorithme de bandit contre un adversaire, afin d'en étudier l'intérêt et les limites.

ABSTRACT

On-line Interactive Learning of Natural Language Neural Generation for Human-machine Dialogue.

Recently, some propositions have emerged to handle natural language generation in spoken dialog systems with recurrent neural network models (Wen *et al.*, 2016a). Those models require a huge amount of learning data, which complicates the data collection and annotation. To address this issue, we propose an online-learning protocol, based on a reinforcement learning approach. We learn a model on a reduced corpus produced with templates, and adapt it online. For this first experiment, we propose an approach using an adversarial bandit algorithm, studying its advantages and limitations.

MOTS-CLÉS : génération en langage naturel, réseau de neurones récurrent, bandit contre un adversaire, adaptation à l'utilisateur.

KEYWORDS: natural language generation, recurrent neural network, adversarial bandit, on-line learning, user adaptation.

1 Introduction

Dans un système de dialogue oral, le composant de génération en langage naturel (NLG pour *Natural Language Generation*) vise à produire une phrase à partir d'un acte de dialogue (AD) du système, proposé par le gestionnaire de dialogue. Par exemple, l'acte $inform(name = bar\_metropol, type = bar, area = north, food = french)$ peut générer la phrase « *Bar Metropol est un bar au nord de la ville qui propose de la cuisine française* ». Pour produire les réponses, les systèmes utilisent traditionnellement des patrons et des règles rédigés par des experts et associés à des ressources linguistiques (Rambow *et al.*, 2001). Ce type de modèles NLG produit des phrases de bonne qualité

mais répétitives et seulement pour des tâches bien spécifiées. Plusieurs approches stochastiques basées sur les données ont conduit à remettre en cause ces méthodes car elles présentent notamment deux défauts : le manque de généralisation pour des domaines ouverts et la répétition fréquente de phrases identiques et de structures rigides.

Oh et Rudnicky ont proposé d'utiliser un modèle de langue n-grammes pour surgénérer un ensemble de phrases candidates, dans lequel est retenue la forme finale (Oh & Rudnicky, 2002). Mairesse et Young ont étendu ce modèle en introduisant des facteurs construits sur une représentation sémantique peu profonde pour construire des n-grammes de segments (Mairesse & Young, 2014). Plus récemment Wen *et al.* ont proposé plusieurs modèles basés sur les RNN (*Recurrent Neural Networks*) (Wen *et al.*, 2015a,c,b). Les évaluations par des humains ont montré que ces systèmes peuvent générer des phrases de bonne qualité linguistique et plus variées que les patrons. L'utilisation de modèles neuronaux récurrents encodeur-décodeur a aussi été étudiée pour construire des systèmes de dialogue de bout en bout (Serban *et al.*, 2016). Dans ce cadre, les RNN se chargent de la NLG mais aussi de la compréhension et du processus de décision ; cette étude est toutefois effectuée dans un contexte différent de cet article en considérant des systèmes conversationnels non dirigés par le but.

D'une manière générale, les modèles stochastiques impliquent un travail important pour la production de corpus pour les nouveaux domaines. Wen *et al.* présentent une approche incrémentale pour gérer l'adaptation de domaine d'un modèle de génération à base de RNN (Wen *et al.*, 2016b). Ils recourent à des données contrefaites synthétisées à partir d'un ensemble hors-domaine pour ajuster leur modèle sur un ensemble réduit de phrases du domaine. Il est aussi envisageable d'employer des méthodes d'extension de corpus (Manishina *et al.*, 2016) mais elles ne permettent pas une adaptation simultanée aux préférences de l'utilisateur.

Dans cet article, nous tentons de réduire le coût de la production de nouvelles données, en adaptant un modèle initial, non pas à une nouvelle tâche mais afin de générer des phrases avec une plus grande diversité. Dans cette logique, une approche par renforcement basée sur un algorithme de bandit contre un adversaire est appliquée (Auer *et al.*, 2002). Si cette approche a déjà été utilisée dans les systèmes de dialogue pour la compréhension du langage (Ferreira *et al.*, 2015, 2016), ici nous proposons un protocole pour adapter le modèle RNN aux nouvelles phrases, différentes de l'ensemble d'apprentissage, en prenant en compte le coût requis de l'utilisateur pour les fournir au système.

2 Le problème de l'interaction en ligne

Les modèles NLG neuronaux peuvent donner de bons résultats (Wen *et al.*, 2015c) mais nécessitent une quantité importante de données d'apprentissage annotées pour pouvoir produire des sorties diversifiées et de bonne qualité. Plusieurs exemples de chaque phrase pour chaque acte de dialogue sont en particulier nécessaires pour entraîner le modèle. Afin de réduire le coût de collecte du corpus, nous proposons dans cet article un protocole d'apprentissage en ligne.

Nous procédons en deux étapes. Tout d'abord, un corpus initial, constitué de références générées par patrons, est utilisé pour entraîner un modèle de génération. Puis, le modèle est utilisé pour générer des phrases, en interaction vocale avec l'utilisateur, auquel il peut demander de produire un énoncé meilleur ou différent. À cette étape du développement, nous considérons l'utilisateur comme averti (il peut être un des développeurs ou un collègue), puisqu'il pourrait être hasardeux de laisser n'importe quel utilisateur accéder à une telle fonctionnalité sans un moyen de contrôle efficace de l'adaptation du modèle. Cette difficulté sera traitée dans de futurs travaux.

Afin de réduire l'effort demandé à l'utilisateur et éviter des actions inutiles, nous proposons d'utiliser un algorithme de bandit contre un adversaire pour décider si le système devrait ou non demander l'expertise de l'utilisateur, selon le gain et le coût estimés de cette action.

Concrètement afin de collecter de nouvelles données pour son modèle, le système va devoir décider à chaque tour de parole s'il : 1. doit demander à l'utilisateur une alternative à sa proposition, 2. peut utiliser la transcription automatique de l'entrée utilisateur ou demander un traitement correctif additionnel.

2.1 Cas statique

Une fois que le système génère l'énoncé, le système peut choisir une action (à partir d'une distribution de probabilité) parmi un ensemble $\mathcal{I}$ de M actions. Dans une configuration préliminaire, nous prenons $M = 3$ et $\mathcal{I}$ est défini comme : $\mathcal{I} := \{\text{Skip, AskDictation, AskTranscription}\}$. Soit $i \in \mathcal{I}$ l'indice de l'action. Nous supposons que l'effort de l'utilisateur $\phi(i) \in \mathbb{R}^+$, peut être associé au temps nécessaire pour effectuer l'action i. Les **efforts estimés** associés à chaque action sont :

• **Skip** : n'appliquer aucune mise à jour au modèle. Le coût de cette action est nul ($\phi(\text{Skip}) = 0$).
• **AskDictation** : affiner le modèle en considérant un énoncé alternatif proposé par l'utilisateur et transcrit automatiquement par un système de reconnaissance ($\phi(\text{AskDictation}) = 1$).
• **AskTranscription** : demander à l'utilisateur de transcrire la correction ou l'énoncé alternatif ($\phi(\text{AskTranscription}) = 1 + l$, avec l la taille de l'énoncé proposé).

Nous estimons ensuite le **gain** de l'action choisie $g(i) \in [0, 1]$ comme suit :

• **Skip** : rien n'est appris, le gain est de 0 ($g(\text{Skip}) = 0$).
• **AskDictation** : le gain est calculé en considérant la marge restante du score BLEU (Papineni *et al.*, 2002) de l'énoncé généré par le système, en utilisant l'énoncé proposé par l'utilisateur comme référence $\text{BLEU}_{gen/prop}$. Pour tenir compte des erreurs potentielles ajoutées par le système de reconnaissance vocal, cette mesure est pénalisée par l'estimation d'un WER et d'un SER :

$$g(\text{AskDictation}) = (1 - \text{BLEU}_{gen/prop}) \times (1 - \text{WER}) \times (1 - \text{SER})$$

Le WER (*Word Error Rate*) donne le nombre moyen d'erreurs de transcription au niveau des mots, tandis que le SER (*Slot Error Rate* (Wen *et al.*, 2015c)) évalue la présence des slots (concepts, valeurs) de l'AD et pénalise les énoncés générés qui ne contiennent pas les informations sémantiques requises.
• **AskTranscription** : cette action suppose de demander à l'utilisateur de transcrire manuellement l'énoncé pour éviter les erreurs de transcription automatique. Donc, l'estimation du gain ne tient compte que du score BLEU de l'énoncé généré par le système, en utilisant la phrase proposée par l'utilisateur comme référence ($g(\text{AskTranscription}) = 1 - \text{BLEU}_{gen/prop}$).

Enfin, une fonction de perte $l(i) \in [0, 1]$ est définie afin que le système maximise la mesure de gain $g(i)$ et minimise l'effort de l'utilisateur $\phi(i)$:

$$l(i) = \underbrace{\alpha(1 - g(i))}_{\text{amélioration du système}} + \underbrace{(1 - \alpha)\frac{\phi(i)}{\phi_{max}}}_{\text{effort de l'utilisateur}} \quad (1)$$

α est un scalaire qui pondère le gain par rapport au coût, permettant au système de s'adapter aux préférences de l'utilisateur. ϕ_{max} correspond à l'effort maximal (ici il peut être atteint par AskTranscription et vaut 41, la taille maximale d'une phrase étant fixée à 40 mots) pour normaliser l'effort de l'utilisateur entre 0 et 1.

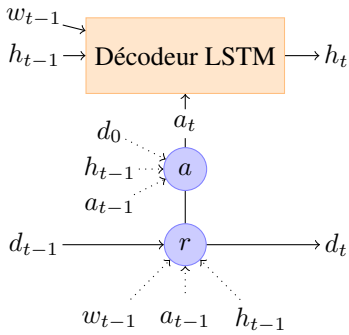


FIGURE 1 – LSTM à contexte combiné

2.2 Cas du bandit contre un adversaire

Nous considérons le scénario suivant pour le problème de bandit contre un adversaire : à chaque itération, le système produit une phrase puis choisit une action $i_t \in \mathcal{I}$. Une fois que l'action i_t est effectuée, le système calcule l'estimation du gain $g(i_t)$, l'effort de l'utilisateur $\phi(i_t)$ et la perte $l(i_t)$. Le but de l'algorithme de bandit est alors de trouver $i_1, i_2, \ldots$, afin que pour chaque t, le système minimise la perte $l(i_t)$.

Nous découpons l'apprentissage en bloc de n itérations chacun. À la fin de chaque bloc, nous ajoutons les énoncés proposés par l'utilisateur à notre corpus d'apprentissage et nous mettons à jour le modèle sur ce corpus étendu. En même temps, nous calculons la fonction de perte pour chaque choix de bandit et nous mettons à jour sa politique. Cette mise à jour du modèle NLG et de la politique décidant de l'action au bout de plusieurs itérations, plutôt qu'à chaque énoncé collecté, permet de réduire le temps de mise à jour du modèle et d'améliorer sa robustesse.

3 Expériences

3.1 Le système de génération

Le modèle NLG utilisé est le LSTM (*Long short-term memory*) à contexte combiné décrit plus en détail dans (Riou *et al.*, 2017). Ce modèle combine deux modèles neuronaux proposés par Wen *et al.*, le modèle LSTM conditionné sémantiquement (SCLSTM) et l'encodeur-décodeur RNN avec attention. Chacun de ces systèmes traite différemment l'information sémantique représentée par un AD pour produire la phrase. La cellule de contrôle (*reading gate*) du SCLSTM permet de filtrer l'AD en ne conservant à chaque étape que l'information restante qui n'a pas été encore traitée. Au contraire, le mécanisme d'attention permet de sélectionner dans l'AD au complet l'information à considérer spécifiquement à l'étape suivante, mais perd la progression dans l'information déjà traitée. L'objectif de notre système est de combiner les avantages des deux systèmes, en utilisant une cellule de contrôle r et un mécanisme d'attention a pour traiter séquentiellement l'AD d_t restant à générer (Figure 1), en prenant en compte l'information h_{t-1} portée dans la couche cachée du LSTM, le mot précédemment généré w_{t-1}, l'attention portée a_{t-1} à un temps donné et l'information d_0 de l'AD en cours de lecture. Ainsi, ce modèle permet de sélectionner dans l'information encore non traitée, celle à considérer spécifiquement à l'étape suivante. Cette information sélectionnée est ensuite envoyée dans un décodeur LSTM pour produire le mot suivant.

3.2 Cadre expérimental

Les expériences ont été menées sur le corpus *SF restaurant* décrit dans (Wen *et al.*, 2015c) et librement accessible en ligne [1]. Ces données contiennent 5 191 phrases, pour 271 actes de dialogue (AD) distincts. Le corpus associe à chaque acte une phrase générée par patron et plusieurs phrases proposées par des annotateurs humains, chaque phrase étant délexicalisée [2].

Le LSTM à contexte combiné a été implémenté en utilisant la bibliothèque Tensorflow [3]. Ce système a ensuite été entraîné sur le corpus séparé en 3 parties suivant un ratio 3 :1 :1 : apprentissage, validation et test, en utilisant uniquement les références proposées par les annotateurs humains.

3.3 Comparaison des systèmes de génération

Une évaluation a été conduite avec deux métriques objectives calculées à partir des générations de référence, le score BLEU-4 (Papineni *et al.*, 2002) et le taux d'erreur SER. Le BLEU valide la génération de phrases, notamment la grammaticalité, tandis que le SER se concentre spécifiquement sur le contenu sémantique. Pour chaque exemple, nous produisons 20 hypothèses et ne gardons pour l'évaluation que les 5 meilleures selon le score donné par le modèle NLG. Des références multiples pour chaque AD sont obtenues en groupant les phrases délexicalisées du même AD et en les relexicalisant ensuite.

Le LSTM à contexte combiné obtient un score BLEU de 71,1%, voisin des deux autres systèmes initiaux (72,2% pour le SCLTSM et 69,7% pour l'encodeur-décodeur), mais le taux d'erreur SER est divisé par trois par rapport aux autres systèmes, en passant de 0,77% et 0,65% pour le SCLSTM et l'encodeur-décodeur, à 0,24% pour le LSTM à contexte combiné. Cela veut dire qu'il propose une meilleure couverture des concepts à exprimer et donc moins d'omissions ou d'erreurs de concepts, ce qui est le principal but recherché pour un module NLG.

3.4 Évaluation de l'apprentissage en ligne

Nous avons utilisé le même corpus, mais avec un découpage en apprentissage, validation et test suivant un ratio 2 :1 :1. Le modèle NLG utilisé est encore le LSTM à contexte combiné. Pour initialiser le modèle, nous l'entraînons en utilisant les références générées par patron du corpus d'apprentissage. Le corpus de validation a permis de décider l'arrêt de la phase d'apprentissage (*early stopping*). Ensuite, nous avons simulé un apprentissage en ligne en réutilisant le corpus d'apprentissage, mais cette fois en apprenant sur les références proposées par des annotateurs humains. Le modèle ainsi que la politique de bandit ont été mis à jour toutes les 400 phrases. Le WER a été simulé en insérant de manière aléatoire des erreurs (confusion, insertion et suppression) dans les exemples du corpus, jusqu'à atteindre un taux global de WER prédéfini.

Le modèle initial, entraîné sur les références générées par patron, atteint un haut score BLEU de 80,2% quand celui-ci est calculé à partir de références générées par patrons, mais qui est réduit à seulement 39,7% en refaisant les calculs à partir des seules références proposées par des annotateurs humains. Cela montre bien la grande diversité possible des réponses dans une conversation.

La Figure 2 montre l'évolution du score BLEU en fonction du coût d'apprentissage de la politique du bandit, avec un WER simulé (fixé à 5%). Le score BLEU est obtenu en testant le modèle sur les

1. https://www.repository.cam.ac.uk/handle/1810/251304
2. La délexicalisation remplace les formes de surface des concepts par des variables, *inform(name=la mimosa, food=mediterranean)* devenant « *$name sert des plats $food* ».
3. https://www.tensorflow.org

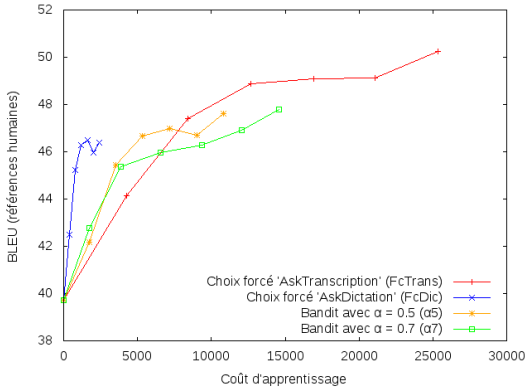
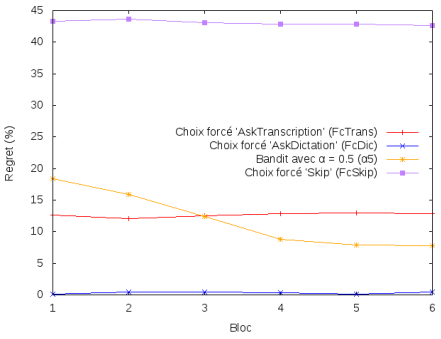


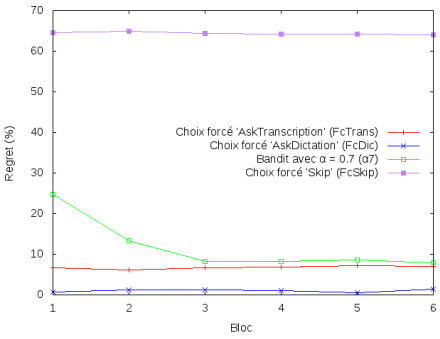
FIGURE 2 – Évolution du score BLEU en fonction du coût cumulé.

références du corpus de test proposées par des annotateurs humains, tandis que le coût d'apprentissage est calculé en sommant les coûts de chaque choix effectué par le bandit durant l'apprentissage en ligne. La taille d'un bloc a été fixée à 400 itérations. Nous avons testé deux configurations limites en forçant le choix 'AskDictation' (FcDic) et en forçant le choix 'AskTranscription' (FcTrans). De plus, nous avons testé le bandit avec deux valeurs différentes pour α : 0,5 ($\alpha 5$) et 0,7 ($\alpha 7$). La seconde valeur réduit l'influence du coût d'apprentissage, permettant au système de demander plus d'effort à l'utilisateur. Chaque courbe est composée de sept points. Le premier correspond au score BLEU du modèle par rapport aux références proposées par des annotateurs humains, avant l'apprentissage en ligne. Les six suivants sont calculés après chaque bloc de 400 phrases. Le coût est cumulé sur tous les blocs précédents. Nous pouvons observer que le bandit réduit avec succès le coût d'apprentissage jusqu'à une certaine quantité de données apprises ; après un coût cumulé de 7 500, AskTranscription devient la meilleure configuration. Une fois toutes les données d'apprentissage utilisées, $\alpha 5$ et $\alpha 7$ atteignent tous deux 47,6% de BLEU, entre les 46,4% pour FcDic et 50,3% pour FcTrans. Le coût pour AskTranscription est bien plus élevé que pour AskDictation. Par conséquent, le bandit apprend mieux au départ que FcTrans, en équilibrant entre les deux choix ; après les deux premiers blocs, l'augmentation du score BLEU ralentit, jusqu'à ce que les courbes d'$\alpha 5$ et $\alpha 7$ passent toutes les deux sous celle FcTrans. Une plus grande valeur d'α tend à favoriser les actions de type Ask sur Skip, et AskTranscription sur AskDictation, ce qui permet notamment de s'adapter aux préférences de l'utilisateur, y compris en cours d'utilisation.

Une autre façon de visualiser l'apprentissage du bandit est d'observer la variation du regret, présentée dans la Figure 3. Le regret représente la marge du bandit pour améliorer son apprentissage. Pour cela, on considère la récompense estimée du bandit ($r(i) = 1,0 - l(i)$). On peut alors calculer le regret comme la différence entre la meilleure récompense possible parmi les trois choix $r_{opti}(i)$ et la récompense estimée $r(i)$. Les six points des courbes correspondent à la somme des regrets sur chaque bloc, exprimée en pourcentage de la somme des récompenses optimales. Nous pouvons observer que le bandit diminue effectivement son regret au cours de l'apprentissage, ce qui vérifie bien l'apprentissage de sa politique. Il n'arrive cependant pas à passer en dessous des 10% de regret. On peut aussi observer que le choix AskDictation possède un très faible regret, ce qui est attendu, puisqu'il permet d'obtenir bien souvent des phrases correctes en demandant moins d'effort à l'utilisateur. Mais il faut noter aussi un effet de moyenne qui cache les exemples possédant beaucoup d'erreurs de transcription, et donc un fort regret. Pour permettre au bandit de détecter ces exemples



(a) $\alpha = 0,5$



(b) $\alpha = 0,7$

FIGURE 3 – Évolution du regret lors de 2 apprentissages simulés, avec $\alpha = 0,5$ et $0,7$.

où le choix AskDictation est à éviter, mais aussi de mieux savoir lorsqu'il est plus intéressant de ne rien demander à l'utilisateur, il devient nécessaire de prendre en compte le contexte de l'exemple traité dans la décision du bandit. Ainsi, tout en gardant le même protocole d'apprentissage en ligne et ses avantages (équilibrage entre différents choix et adaptation à l'utilisateur), nous espérons pouvoir améliorer nos résultats en prenant en compte le contexte de la phrase générée (estimation du nombre d'erreurs lors de la génération, diversité des hypothèses générées par le système NLG) à l'aide d'un bandit contextuel (Auer *et al.*, 2002).

4 Conclusions et perspectives

Dans cet article, nous avons proposé un nouveau protocole pour adapter un modèle initial de génération de langage naturel neuronal à l'aide d'un apprentissage en ligne. Dans une expérience simulée, un algorithme de bandit a permis d'équilibrer de manière automatique l'évolution des performances du système avec le coût induit pour l'utilisateur. Ces résultats ne sont encore que préliminaires et le travail doit être poursuivi. Tout d'abord, les capacités d'apprentissage du bandit doivent être augmentées par la prise en compte du contexte lors de ses décisions, à l'aide d'un bandit contextuel. Une évaluation avec des utilisateurs doit également être menée pour avoir une évaluation plus fiable des coûts et montrer l'intérêt pratique de notre approche.

Remerciements

Ce travail a été partiellement financé par le Labex BLRI (ANR-11-LABX-0036).

Références

AUER P., CESA-BIANCHI N., FREUND Y. & SCHAPIRE R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, **32**(1), 48–77.

FERREIRA E., JABAIAN B. & LEFÈVRE F. (2015). Zero-shot semantic parser for spoken language understanding. In *INTERSPEECH*.

FERREIRA E., REIFFERS-MASSON A., JABAIAN B. & LEFÈVRE F. (2016). Adversarial bandit for online interactive active learning of zero-shot spoken language understanding. In *ICASSP*.

MAIRESSE F. & YOUNG S. (2014). Stochastic language generation in dialogue using factored language models. *Computational Linguistics*, **40**(4), 763–799.

MANISHINA E., JABAIAN B., HUET S. & LEFÈVRE F. (2016). Automatic corpus extension for data-driven natural language generation. In *LREC*.

OH A. H. & RUDNICKY A. I. (2002). Stochastic natural language generation for spoken dialog systems. *Computer Speech & Language*, **16**(3–4), 387–407.

PAPINENI K., ROUKOS S., WARD T. & ZHU W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. In *ACL*.

RAMBOW O., BANGALORE S. & WALKER M. (2001). Natural language generation in dialog systems. In *HLT*.

RIOU M., JABAIAN B., HUET S. & LEFÈVRE F. (2017). Online adaptation of an attention-based neural network for natural language generation. In *INTERSPEECH*.

SERBAN I. V., SORDONI A., BENGIO Y., COURVILLE A. & PINEAU J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence*.

WEN T.-H., GAŠIĆ M., MRKŠIĆ N., BARAHONA L. M. R., SU P.-H., ULTES S., VANDYKE D. & YOUNG S. (2016a). Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*.

WEN T.-H., GAŠIĆ M., KIM D., MRKŠIĆ N., SU P.-H., VANDYKE D. & YOUNG S. (2015a). Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGDIAL*.

WEN T.-H., GAŠIĆ M., MRKŠIĆ N., ROJAS-BARAHONA L. M., SU P.-H., VANDYKE D. & YOUNG S. (2015b). Toward multi-domain language generation using recurrent neural networks. In *NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*.

WEN T.-H., GAŠIĆ M., MRKŠIĆ N., ROJAS-BARAHONA L. M., SU P.-H., VANDYKE D. & YOUNG S. (2016b). Multi-domain neural network language generation for spoken dialogue systems. In *NAACL-HLT*.

WEN T.-H., GAŠIĆ M., MRKŠIĆ N., SU P.-H., VANDYKE D. & YOUNG S. (2015c). Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *EMNLP*.