Architecture basée sur les mécanismes d'attention: le cas de la génération neuronale de questions

Thomas Scialom[**,***] — **Benjamin Piwowarski**[*,**] — **Jacopo Staiano**[***]

[*] *CNRS*
[**] *LIP6 - Sorbonne Universités, UPMC Univ Paris 06, UMR 7606*
prénom.nom@lip6.fr
[***] *reciTAL, 1 rue d'Hauteville 75010 Paris,*
prénom@recital.ai

RÉSUMÉ. *Les architectures neuronales basées sur l'attention, telles que le* Transformer*, ont récemment suscité l'intérêt de la communauté scientifique et ont permis d'obtenir des progrès importants par rapport à l'état de l'art dans plusieurs domaines. L'adaptation des Transformers à la tâche de la génération de questions n'est pas simple car les données sont ici relativement peu volumineuses. Nous explorons, par conséquent, comment un Transformer peut être adapté et, en particulier, étudions l'effet des mécanismes de copie, de remplacement d'entité nommée ainsi que l'intégration de représentations de mots contextualisées. Ces mécanismes sont particulièrement utiles pour le traitement des mots hors vocabulaire, qui sont les plus susceptibles d'affecter les performances dans le cadre de tâches pour lesquelles les données sont relativement moins disponibles. Les expériences rapportées montrent des résultats encourageants dans le scénario où la réponse n'est pas connue (mode non guidé). On obtient, par ailleurs, une amélioration par rapport à l'état de l'art quand elle ne l'est pas (mode guidé).*

ABSTRACT. *Neural architectures based on self-attention, such as Transformers, recently attracted interest from the research community, and obtained significant improvements over the state of the art in several tasks. Adapting Transformers to Neural Question Generation is not straightforward as data is relatively scarce in this task. We hence explore how Transformers can be adapted, and, in particular, study the effect of copy mechanisms, placeholders, and contextual word embeddings. Those mechanisms are particularly useful for the treatment of out-of-vocabulary words, which are more likely to affect performance in tasks with relatively smaller data available. The experiments reported show encouraging results in the* answer-aware *sce-*

nario (for which the target answer is known), while improvements over the state-of-the-art systems are obtained in the answer-agnostic *setup.*

MOTS-CLÉS : *génération de questions, attention*

KEYWORDS: *neural question generation, attention*

1. Introduction

La communauté MRC (Machine Reading Comprehension) se concentre sur le développement de modèles et d'algorithmes permettant aux machines de représenter correctement le sens de phrases en langage naturel, afin de réaliser des tâches de haut niveau, telles que fournir des réponses à des questions, générer des résumés ou encore des questions pertinentes à partir d'un morceau de texte. La performance de ces tâches indique à quel point les différentes architectures proposées sont capables de capturer le sens en langage naturel. Récemment, les architectures neuronales basées sur l'attention ont amélioré les performances de l'état de l'art dans plusieurs tâches telles que la modélisation du langage et la traduction automatique, pour lesquelles des données abondantes sont disponibles. Pour autant, ils n'ont pas fait l'objet d'une évaluation approfondie pour des problèmes où les données sont plus rares. Nous étudions donc ici l'application d'un modèle neuronal qui fait référence dans de nombreuses tâches, les Transformers, à la tâche de la Génération Neuronale de Questions (GNQ) : à partir d'un extrait de texte (et éventuellement d'une réponse cible), le modèle génère des questions correspondant à l'extrait et le cas échéant à la réponse cible en utilisant des mécanismes d'auto-attention – i.e. la séquence de représentations de mots initiale est affinée en plusieurs étapes.

La tâche que nous étudions est celle de la génération de question à partir d'un texte contenant la réponse. Alors que plusieurs travaux ont été menées sur le problème de la génération de questions à l'aide de systèmes basés sur des règles (Rus *et al.*, 2010), la communauté scientifique s'est récemment tournée vers les approches neuronales : a notre connaissance, seul (Du *et al.*, 2017) a proposé une approche seq2seq (séquence à séquence) de bout en bout dans le cas non guidé par la réponse contenue dans le texte d'entrée (mode *non guidé*). Inversement, des modèles de génération de questions axés sur la position de la réponse dans le texte d'entrée (mode *guidé*) ont été proposés par (Zhou *et al.*, 2017) et, plus récemment, (Zhao *et al.*, 2018). Ce dernier atteint l'état de l'art en utilisant un encodeur avec de l'auto-attention et un décodeur avec un mécanisme de copie spécifique (*copy maxout*) permettant de recopier tels quels des mots du texte contenant la réponse. En outre, (Song *et al.*, 2017) a proposé un modèle génératif entraîné conjointement à la génération de questions ainsi que de réponses. Nous notons que tous ces travaux utilisent le jeu de données SQuAD (Rajpurkar *et al.*, 2016), que nous utilisons par conséquent dans toutes les expériences rapportées dans cet article. Pour adapter le Transformer au scénario GNQ et, de façon générale, aux scénarios où les données ne sont pas abondantes, nous complétons l'architecture de base par un mécanisme de copie, un remplacement d'entité nommée et enfin un ajout de représentations contextualisées : ces mécanismes sont particulièrement utiles pour le traitement des mots hors vocabulaire (mots peu ou pas présents dans le corpus d'entraînement). Ce problème est particulièrement susceptible d'affecter les performances dans le cadre de tâches où les données sont rares. Dans cet article, nous étudions l'effet de chacun de ces mécanismes sur des architectures basées sur l'attention. Au delà de la tâche de génération de questions, ces mécanismes peuvent être utilisés dans de nombreux modèles d'accès à l'information basés sur des réseaux de neurones dotés

de mécanismes d'attention, comme par exemple pour de la réponse aux questions ou de la génération de résumé automatique. Nous discutons des améliorations apportées au Transformer, et obtenons un nouvel état de l'art dans en mode *non guidé*, ainsi que des résultats encourageants lorsque la réponse est connue (mode *guidé*). Les résultats obtenus montrent l'importance de ces mécanismes pour des tâches complexes d'accès à l'information.

2. Architecture de base

Les modèles neuronaux de type séquence à séquence (*seq2seq*) s'appuient souvent sur des architectures encodeur-décodeur : une séquence de mot est transformée en une nouvelle séquence de mots. Dans ce cadre, les réseaux de neurones récurrents – qui génèrent ou représentent un texte mot par mot – ont atteint des performances de l'état de l'art pour les tâches de traitement du langage naturel telles que le résumé automatique (Chopra *et al.*, 2016) et la traduction (Sutskever *et al.*, 2014). Parmi les inconvénients des modèles récurrents, on note des obstacles inhérents à la parallélisation avec les coûts en temps de calcul qui en résultent, ainsi que des difficultés pour gérer les dépendances de longue distance au sein d'un texte. Le récent modèle Transformer (Vaswani *et al.*, 2017), qui s'est avéré très efficace pour plusieurs tâches (Devlin *et al.*, 2018 ; Radford *et al.*, 2018), résout ces problèmes en éliminant la récurrence. Il peut être décrit brièvement comme un modèle seq2seq avec un encodeur et un décodeur symétriques reposant sur un mécanisme d'attention.

Plus précisément, étant donné une séquence d'unités linguistiques t (mots ou n-grammes), constituant le *contexte* à partir duquel une question devrait être générée, chaque unité t_i est représentée par un vecteur dense $v_i \in \mathbb{R}^d$ (représentation de dimension d).

Cette représentation est calculée comme la somme de :

1) les représentations (*embeddings*) t_i de l'unité à la position i. Celles-ci peuvent être pré-entraînés via Glove (Pennington *et al.*, 2014), Word2vec (Mikolov *et al.*, 2013) ou ELMO (Peters *et al.*, 2018a), initialisés de manière aléatoire ou une concaténation de différentes représentations ;

2) la représentation de la position p_i (nécessaire pour fournir au Transformer des informations sur l'ordre des unités linguistiques) ; cette représentation est précalculée (Gehring *et al.*, 2017) de façon à ce que le produit scalaire $p_i \cdot p_{i'}$ soit d'autant plus grand que les positions i et i' sont proches, tout en faisant en sorte de se répartir dans l'espace de représentation afin de pouvoir être considérée comme du bruit pour les représentations sémantiques des unités ;

La représentation initiale de chaque unité à la position i est donc définie par :

$$t_i^{(0)} = t_i + p_i$$

L'encodeur, qui permet de représenter le texte contenant la réponse dans notre

cas, est constitué de N blocs identiques d'auto-attention : cela permet d'affiner la représentation de chaque unité (mot ou n-gramme) de la séquence de manière itérative. Chaque bloc d'auto-attention calcule une nouvelle représentation de chaque unité t_i. De manière plus précise, pour le $l$ème bloc, la nouvelle représentation $t_i^{(l+1)}$ de chaque unité i est guidée par un mécanisme d'attention de $t_i^{(l)}$ sur l'ensemble des unités $t_j^{(l)}$:

1) On calcule tout d'abord une *attention* sur les unités $p(j|t_i^{(l)};\theta)$ qui dépend de la représentation actuelle de l'unité à la position j et des paramètres du modèle θ (la somme sur toutes les positions est égale à 1) ;

2) La nouvelle représentation après le bloc l est définie comme une combinaison linéaire des représentations à l'étape l modulée par les attentions

$$t_i^{(l+1)} = \sum_j p(j|t_i^{(l)};\theta) f(t_j^{(l)};\theta)$$

Au bout de N itérations, la représentation finale, $t_j^{(N)}$, de chacune des unités est donc une représentation de l'unité j selon son contexte, i.e. toutes les autres unités de la séquence. L'idée de ces étapes est que la représentation de chaque unité va s'affiner progressivement : par exemple, dans "le chat noir", la représentation du mot "chat" sera pourrait être une combinaison des représentations de "noir" et "chat" – la représentation du mot "chat" à la fin du processus itératif inclut donc des informations issues de son contexte.

Dans le modèle Transformer, le mécanisme d'attention est le résultats de plusieurs attentions effectuées en parallèle sur des parties différentes des représentations (appelées "têtes" d'attention), permettant au modèle de calculer des représentations plus fines de chaque mot.

Le décodeur, qui génère la question dans notre cas, se base sur une séquence partielle (la séquence en train d'être générée), pour prédire l'unité suivante qui sera générée. Le mécanisme est similaire à celui de l'encodeur, avec N_d étapes : soit $s_1, \ldots, s_m$ la séquence qui a été générée jusqu'à l'étape m, nous obtiendrons des représentations finales, $s_1^{(N_d)}, \ldots, s_m^{(N_d)}$ par le biais du mécanisme d'auto-attention. La grande différence avec le processus d'encodage est que ce processus d'attention porte également sur la séquence d'entrée.

Finalement, l'information est résumée dans vecteur unique h_m^* (ou plus simplement h^*) par le biais d'un dernier mécanisme d'attention, i.e.

$$h_m^* = h(\sum_j p(j|s_i^{(N_d)};\theta) g(s_j^{(N_d)});\theta) \quad [1]$$

où h et g sont deux fonctions dont les paramètres sont appris.

Ce vecteur h^* a une dimension fixe, ce qui permet de l'utiliser comme une entrée du module de classification qui prédit la probabilité du mot suivant w de la question générée, i.e.

$$p(w|h^*) \propto \exp\left(h^* \cdot x_w\right)$$

où x_w est un vecteur de paramètres appris pour le mot w.

Notre architecture est basée sur l'implémentation d'origine de (Vaswani *et al.*, 2017) proposée dans la librairie de tensor2tensor[1]. Parce que l'utilisation des Transformers est devenue classique dans la littérature récente et que notre implémentation est identique à l'originale, nous omettons une description plus en détail de l'architecture de ce modèle et proposons au lecteur de se référer à (Vaswani *et al.*, 2017) ainsi que l'excellent guide "The Annotated Transformer."[2]

Dans ce travail, nous avons testé le Transformer avec ses paramètres originels ainsi qu'une version plus simple avec deux têtes d'attention, et une dimension de représentation $H = 256$. Ayant obtenu des résultats similaires avec ces deux versions, les résultats rapportés dans ce papier sont obtenus avec le modèle le plus simple.

3. Expériences

La motivation de nos expériences provient des résultats préliminaires obtenus en appliquant l'architecture de base du Transformer à la tâche GNQ. Dans cet article, nous étudions comment différents mécanismes appliqués à une architecture Transformer contribuent au succès sur des tâches pour lesquelles peu de données sont disponibles, telles que la génération neuronale de questions. Dans ce qui suit, nous décrivons et évaluons l'apport à l'architecture de base de Transformer de : *a)* un mécanisme de copie ; *b)* une stratégie de remplacement d'entité nommée ; et *c)* représentation contextualisée des mots.

3.1. *Données*

Pour nos expériences, nous utilisons le jeu de données Stanford Question Answering (SQuAD) (Rajpurkar *et al.*, 2016) qui fait référence. Il contient plus de 100 000 questions posées par des humains sur des articles choisis de Wikipedia. Nous évaluons les performances à l'aide de la métrique BLEU (Papineni *et al.*, 2002) couramment utilisée et nous les comparons aux modèles GNQ les plus récents. BLEU correspond ici au pourcentage de n-grammes ($n = 2$ pour BLEU-2) de la question générée qui font partie de la référence (une question écrite manuellement). Dans SQUAD, chaque question est associée à la réponse correspondante et au passage qui la contient. Afin de pouvoir comparer équitablement l'approche proposée avec les efforts de recherche mentionnés précédemment, nous utilisons les partitionnements du jeu de données publiés par (Zhou *et al.*, 2017) dans le scénario *guidé*, et ceux de (Du *et al.*, 2017) dans le cas *non guidé*. Pour toutes les expériences, conformément à la plupart des travaux précédents, le contexte permettant de générer une question correspond à une seule phrase.

1. https ://github.com/tensorflow/tensor2tensor
2. http ://nlp.seas.harvard.edu/2018/04/03/attention.html

3.2. *Représentation enrichie de mots*

Pour traiter les mots rares/jamais vus, l'architecture Transformer (Vaswani *et al.*, 2017) exploite de grandes quantités de données et une segmentation en n-grammes de taille variable (e.g. "the_/cat_/is_on_/the_/mat"). Dans le tableau 2, nous observons que les performances obtenues avec un Transformer standard ne sont pas satisfaisantes pour la tâche GNQ. Nous pensons que ceci est du au manque de données disponibles pour entraîner le modèle. Afin de palier à ce problème, nous proposons d'utiliser une représentation pré-apprise de mots. Plus précisément, nous utilisons une segmentation au niveau des mots, et les représentations de mots GloVe (Pennington *et al.*, 2014) qui sont pré-entraînées [3]. Cette représentation est dite non contextualisées car la représentation de chaque mot ne dépend pas du contexte de la phrase dans la quelle il se trouve. Ainsi, le mot *signes* dans *il parle la langue des signes* et *Tu signes de la main gauche* aura la même représentation.

De plus, comme (Chen et Manning, 2014), pour chaque mot t_i, nous concaténons la représentation du mots v_i avec des caractéristiques supplémentaires f_i, qui dans notre cas correspondent aux catégories grammaticales. Dans le cas *guidé*, comme (Zhou *et al.*, 2017), f_i comporte une caractéristique binaire supplémentaire indiquant pour chaque unité de la séquence s'il appartient à la réponse a. Dans le reste du document, nous nous référons à cette configuration, servant d'architecture de base pour nos expériences, par *Transformer_base*.

Utiliser une représentation pré-apprise au niveau des mots entraîne le problème des mots inconnus lors de la phase de test, pour lequel l'utilisation de n-grammes était une solution. Dans la suite, nous proposons deux solutions permettant de réduire l'importance du problème des mots inconnus, afin de conserver les avantages des représentations pré-apprises sur de grands corpus de données.

3.3. *Stratégie de substitut (placeholder)*

Une méthode permettant au modèle de traiter les mots rares/non vus consiste à remplacer des mots spécifiques par des mots-clés réservés. Un tel mécanisme est souvent utilisé par l'industrie dans les systèmes de traduction neuronale (Crego *et al.*, 2016 ; Levin *et al.*, 2017) pour permettre la copie des entités nommées de la langue source à la langue cible. Comme les entités nommées[4] font également partie des mots rares/jamais vus, nous recourons à cette stratégie et les remplaçons par des mots "virtuels" fixes : tous les mots du contexte reconnus comme entité nommée par le modèle de NER sont remplacés par un mot indiquant leur type et leur ordre d'apparition. Par exemple, *Nikola Tesla est né en 1856.* devient *Personne_1 Personne_2 est née à Date_1*. Durant l'entraînement la même procédure est appliquée aux questions

3. `http://nlp.stanford.edu/data/glove.840B.300d.zip`

4. Au cours de nos expériences, nous avons utilisé la bibliothèque `spaCy 2.0` (`http://spacy.io`) pour le NER, le POS et la segmentation.

	BLEU1	**BLEU2**	**BLEU3**	**BLEU4**	*% copiés*
Vanilla Transformer	36.13	17.77	10.04	6.04	*4.2*
Transformer_base	38.74	20.54	12.26	7.66	*5.7*
+Copie (C)	39.81	22.47	14.25	9.32	*9.1*
+ELMO (E)	40.44	23.87	15.74	10.62	*6.5*
+C+E	41.72	25.07	16.77	11.58	*10.4*
*+Substitut (**P**laceholder)*	41.54	25.52	17.56	12.49	*48.4*
+P+E	42.2	26.2	18.14	12.92	*49.4*
+P+C	42.72	26.52	18.28	13.0	*50.9*
+P+C+E	**43.33**	**26.27**	**18.32**	**13.23**	*51.7*
(Du *et al.*, 2017)	43.09	25.96	17.50	12.28	-

Tableau 1. *Comparaison avec l'état de l'art dans le mode* non guidé *; L'architecture* Transformer_base*, à partir de laquelle tous les modèles sous-jacents sont construits, segmente le texte en mots, et utilise des représentations GloVe pré-entraînées, plutot que une segmentation variable (word pieces), comme pour* Transformer *originel. Nous rapportons également le ratio d'unités générés par les modèles par rapport au nombre d'unités OOV/entité nommée communs entre la source et la cible.*

cibles ; au moment de l'inférence, les espaces réservés sont remplacés par les entités nommées correspondantes comme étape de post-traitement. Cela signifie qu'un vecteur différent, initialisé de manière aléatoire, et qui est appris durant l'entraînement, est utilisé en tant que représentation pour chaque mot-clef de substitution, à la place de la représentation GloVe correspondant au mot d'origine (ou au mot virtuel OOV/<unk> pour les mots qui ne font pas partie du vocabulaire choisi).

Comme indiqué dans la ligne (P) du tableau 1, ce mécanisme seul permet à l'architecture *Transformer_base* d'obtenir des résultats à l'état de l'art. En outre, il s'agit de l'amélioration relative la plus importante par rapport à l'architecture de base, ce qui peut s'expliquer par la nature du jeu de données SQuAD, dans lequel plus de 50% des réponses sont des entités nommées (voir le tableau 2 dans (Rajpurkar *et al.*, 2016)). Ceci est en accord avec le pourcentage de mots copiés par le seul mécanisme de replacement d'entité nommée que nous rapportons ; de plus, cela permet une réduction significative de la taille du vocabulaire ($\sim 30\%$). Néanmoins, une limite de cette stratégie provient de l'utilisation d'un modèle de NER : lorsque celui-ci ne reconnaît pas une entité, la substitution n'a aucun effet, ce qui est particulièrement préjudiciable lorsqu'un mot n'était pas assez fréquent pour être inclus dans le vocabulaire.

3.4. *Mécanisme de copie*

Comme les questions générées à partir d'un contexte donné ont tendance à faire référence à des phrasés spécifiques ou à des entités y apparaissant, (Gulcehre *et al.*, 2016) propose d'utiliser un mécanisme de pointeur (appelé *pointer-softmax*)

pour sélectionner les mots à copier de la phrase source ; intuitivement, cette méthode est particulièrement utile dans le cas de mots rares ou inconnus, car plutôt que de devoir prédire ce mot parmi l'ensemble du vocabulaire, il suffit de savoir quel mot copier parmi les entrées.

Pour utiliser ce mécanisme, nous définissons la probabilité de copier une unité $p_{gen} \in [0, 1]$ au temps t de la façon suivante :

$$p_{gen} = \sigma(W \cdot (h^* \oplus s_t \oplus x_t)) \quad [2]$$

où W est un vecteur de paramètres à apprendre, h^* est donné dans l'équation [1], s_t est décrit en section 2 et x_t sont les unités en entrée du décodeur (*i.e. la représentation GloVe du mot précédemment généré, "<SOS>" pour start of sentence, lorsque c'est le premier mot de la séquence*).

Nous avons testé plusieurs mécanismes d'attention pour permettre la copie, notamment l'attention globale proposée par (Luong *et al.*, 2015) ; comme aucune différence significative n'a été observée, nous avons utilisé, pour nos expériences, les scores d'attention bruts du Transformer sur l'entrée, évitant ainsi l'ajout de nouveau paramètres.

Les résultats de la ligne (C) rapportés dans le tableau 1 montrent comment l'ajout de la copie améliore les performances du modèle. Le mécanisme permet d'augmenter le nombre de mots copiés et de compléter les mécanismes de substitution d'entité nommée lorsque celles-ci ne sont pas correctement reconnues.

L'exemple suivant tiré de SQuAD illustre la contribution du mécanisme de copie : étant donné le contexte, *Beyoncé attended St. Mary's elementary school in Fredericksburg, Texas, where she enrolled in dance classes*, le NER ne parvient pas à reconnaître *Beyoncé* en tant qu'entité nommée. De plus, *Beyoncé* n'est pas dans le vocabulaire. Ainsi le Transformer + substitut produit *where did madonna attend ... ?* tandis que l'ajout du mécanisme de copie permet de récupérer correctement la bonne entité et permet au modèle de générer une question correcte : *where did beyoncé attend ... ?*.

3.5. *Représentations contextualisées*

Les approches de représentation contextualisées permettent de représenter un mot donné en fonction du contexte dans lequel il apparaît, par opposition aux vecteurs fixes et sans contexte fournis par GloVe, permettant ainsi de capturer plus d'informations pour les mots virtuels OOV (Out-Of-Vocabulary). La stratégie de substitution décrite ci-dessus présente l'inconvénient de priver la représentation textuelle en entrée de toute information sémantique en dehors du type d'entité. Par exemple, deux entités telles que Tesla et Edison pourraient avoir des représentations proches dans l'espace vectoriel sémantique, au sein d'un sous-ensemble de mots à caractère scientifique : l'utilisation d'un mot-clef de remplacement empêche ainsi l'utilisation de telles informations. Afin d'étudier l'importance de ce problème, pour chaque mot, nous concaténons les vecteurs sans contexte (voir 3.2) avec la représentation ELMO (Peters *et al.*, 2018b) correspondante au stade de l'encodage. ELMo est une représentation

vectorielle des mots qui varie en fonction du contexte de la phrase ; ils correspondent aux états les plus profonds d'un modèle de neurones récurrent, pré-entraîné sur un grand corpus de texte. Dans nos expériences, celles-ci ne sont utilisées qu'au stade de l'encodage puisqu'elles n'ont pas de sens lorsqu'elles sont appliquées à des phrases incomplètes (i.e. durant la génération).

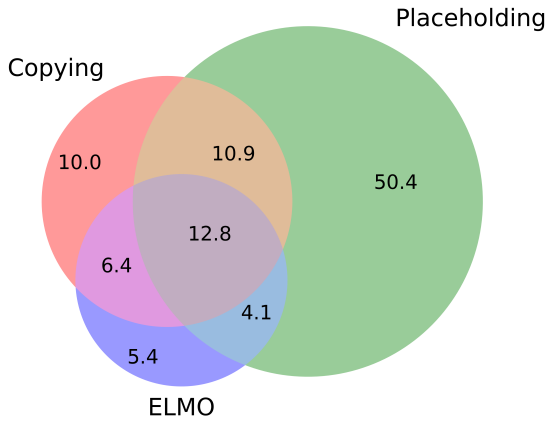

Figure 1. *Pourcentage de mots OOV copiés par les différents mécanismes et leurs combinaisons sur tous les mots OOV copiés.*

Combinées au mécanisme décrit précédemment, les représentation contextualisées permettent d'améliorer encore les performances en obtenant un score BLEU4 de 13,23, presque un point absolu au-dessus de l'état de l'art dans la tâche *réponse-agnostique*.

Dans la figure 1, nous observons dans un diagramme de Venn le pourcentage de mots en dehors du vocabulaire (OOV) copiés par chacun des trois mécanismes. Comme pour la performance BLEU, c'est le mécanisme de substitut qui est le principal contributeur (50% des mots OOV sont uniquement générés pour ce modèle). Cependant chacun des mécanismes permet de copier des mots OOV que les autres mécanismes n'ont pas copiés : 10% pour le mécanisme de copie et 5% pour ELMO, et seulement 13% des mots sont communs aux trois mécanismes.

4. Conclusions et travaux à venir

Nous avons décrit une étude préliminaire sur l'adaptation des architectures du Transformer à des tâches où les données disponibles pour l'entraînement du modèle sont rares – ici, la génération neuronale de questions. Les résultats obtenus montrent la contribution de techniques auxiliaires qui se complètent telles que le mécanisme de copie, la substitution d'entités nommées et les représentations contextualisés. La meilleure performance est obtenue en utilisant ensemble les trois mécanismes, avec

une amélioration de presque un point BLEU4 par rapport à l'état de l'art dans la tâche *non guidée*.

	BLEU4
(Yuan *et al.*, 2017)	10.5
(Duan *et al.*, 2017)	12.28
(Zhou *et al.*, 2017)	13.29
(Yao *et al.*, 2018)	13.36
(Zhao *et al.*, 2018)	**15.32**
Notre modèle	**13.63**

Tableau 2. *Comparaison avec l'état de l'art dans le scénario* guidé*. Notre modèle fait référence à la configuration la plus performante dans la tâche* guidé *ci-dessus.*

D'autre part, pour la tâche *guidée*, nous observons que la même architecture n'atteint pas les performances les plus récentes, bien que les résultats soient encourageants, comme le montre le tableau 2. Bien que l'étude des facteurs expliquant ces résultats fasse partie des travaux futurs, nous notons que pour (Zhao *et al.*, 2018) (système le plus performant), le mécanisme de copie apporte une amélioration significative de près de 3,7 points BLEU4, alors que dans notre cas seul un gain de performance mineur est observé. Les travaux futurs incluent également le test de différentes stratégies pour permettre l'utilisation d'intégrations de représentations contextualisées tels que ELMO (Peters *et al.*, 2018b), plus seulement dans l'encodeur, mais également lors du décodage. De plus, nous allons cibler différentes tâches similaires quant à la rareté des données et évaluer la cohérence des mécanismes mis en place.

5. Bibliographie

Chen D., Manning C., « A fast and accurate dependency parser using neural networks », *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, p. 740-750, 2014.

Chopra S., Auli M., Rush A. M., « Abstractive sentence summarization with attentive recurrent neural networks », *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 93-98, 2016.

Crego J., Kim J., Klein G., Rebollo A., Yang K., Senellart J., Akhanov E., Brunelle P., Coquard A., Deng Y. *et al.*, « SYSTRAN's Pure Neural Machine Translation Systems », *arXiv preprint arXiv :1610.05540*, 2016.

Devlin J., Chang M.-W., Lee K., Toutanova K., « Bert : Pre-training of deep bidirectional transformers for language understanding », *arXiv preprint arXiv :1810.04805*, 2018.

Du X., Shao J., Cardie C., « Learning to Ask : Neural Question Generation for Reading Comprehension », *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, vol. 1, p. 1342-1352, 2017.

Duan N., Tang D., Chen P., Zhou M., « Question generation for question answering », *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 866-874, 2017.

Gehring J., Auli M., Grangier D., Yarats D., Dauphin Y. N., « Convolutional sequence to sequence learning », *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, p. 1243-1252, 2017.

Gulcehre C., Ahn S., Nallapati R., Zhou B., Bengio Y., « Pointing the Unknown Words », *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, vol. 1, p. 140-149, 2016.

Levin P., Dhanuka N., Khalil T., Kovalev F., Khalilov M., « Toward a full-scale neural machine translation in production : the Booking. com use case », *arXiv preprint arXiv :1709.05820*, 2017.

Luong T., Pham H., Manning C. D., « Effective Approaches to Attention-based Neural Machine Translation », *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412-1421, 2015.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient estimation of word representations in vector space », *arXiv preprint arXiv :1301.3781*, 2013.

Papineni K., Roukos S., Ward T., Zhu W.-J., « BLEU : A Method for Automatic Evaluation of Machine Translation », *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 311-318, 2002.

Pennington J., Socher R., Manning C. D., « GloVe : Global Vectors for Word Representation », *Empirical Methods in Natural Language Processing (EMNLP)*, p. 1532-1543, 2014.

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., « Deep contextualized word representations », *arXiv preprint arXiv :1802.05365*, 2018a.

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., « Deep contextualized word representations », *Proc. of NAACL*, 2018b.

Radford A., Narasimhan K., Salimans T., Sutskever I., « Improving language understanding by generative pre-training », *URL https ://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*, 2018.

Rajpurkar P., Zhang J., Lopyrev K., Liang P., « SQuAD : 100,000+ Questions for Machine Comprehension of Text », *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 2383-2392, 2016.

Rus V., Wyse B., Piwek P., Lintean M., Stoyanchev S., Moldovan C., « The First Question Generation Shared Task Evaluation Challenge », *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 251-257, 2010.

Song L., Wang Z., Hamza W., « A Unified Query-based Generative Model for Question Generation and Question Answering », *arXiv preprint arXiv :1709.01058*, 2017.

Sutskever I., Vinyals O., Le Q. V., « Sequence to Sequence Learning with Neural Networks », *Proc. NIPS*, Montreal, CA, 2014.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. u., Polosukhin I., « Attention is All you Need », *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., p. 5998-6008, 2017.

Yao K., Zhang L., Luo T., Tao L., Wu Y., « Teaching Machines to Ask Questions. », *IJCAI*, p. 4546-4552, 2018.

Yuan X., Wang T., Gulcehre C., Sordoni A., Bachman P., Zhang S., Subramanian S., Trischler A., « Machine Comprehension by Text-to-Text Neural Question Generation », *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Association for Computational Linguistics, p. 15-25, 2017.

Zhao Y., Ni X., Ding Y., Ke Q., « Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks », *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3901-3910, 2018.

Zhou Q., Yang N., Wei F., Tan C., Bao H., Zhou M., « Neural question generation from text : A preliminary study », *National CCF Conference on Natural Language Processing and Chinese Computing*, Springer, p. 662-671, 2017.