Data-to-Text: Vers la génération de texte à partir de données non-linguistiques

Clément Rebuffel

Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, F-75005 Paris, France
BNP PARIBAS - Analytics Consulting - Data & AI Lab

RÉSUMÉ. *Nous nous intéressons à la problématique de la génération du langage naturel dont l'objectif est de transcrire un contexte d'entrée vers une description adéquate de ce contexte. Plus particulièrement, nous abordons la problématique du "data-to-text" qui se focalise sur les descriptions de données non linguistiques, comme les tableaux numériques ou les graphiques. Dans ce papier, nous exposons l'état de l'art relatif à ce domaine : nous décrivons les mécanismes de base de la traduction neuronale automatique (NMT) qui sont les fondements des modèles de génération et les avancées récentes pour le cas particulier du "data-to-text". Enfin, nous discutons des nouvelles problématiques dans ce domaine et présentons notre positionnement pour les prochaines recherches de thèse.*

ABSTRACT. *We are interested in the issue of natural language generation, which aims to transcribe an entry context into an adequate description of this context. In particular, we address the issue of data-to-text, which focuses on descriptions of non-linguistic data, such as numerical tables or graphs. In this paper, we survey the state of the art in this field: we describe the core mechanisms of automatic neural translation (NMT) which are the foundations of NLG systems and recent advances in the particular case of data-to-text. Finally, we discuss new issues in this field and present our positioning for future thesis research.*

MOTS-CLÉS : *Génération de langage naturel, Réseau de neurones profond, data-to-text.*

KEYWORDS: *Natural language generation, Deep neural network, Data-to-text.*

1. Introduction

La génération en langage naturel (*"Natural Language Generation"* - NLG) est la tâche qui vise à produire automatiquement un texte compréhensible en langage naturel (Reiter et Dale, 2000). Typiquement, les systèmes de NLG s'appuient sur une représentation arbitraire en entrée et utilise leurs connaissances du langage et du domaine pour proposer un texte, un message d'aide, des explications, etc. Historiquement, ce sont les problématiques *texte-vers-texte*, dont l'objectif est de produire un texte (e.g., résumé) à partir d'un autre texte, qui ont été étudiées. On dénombre de nombreuses applications autour de cette tâche telles que la traduction (Sutskever *et al.*, 2014), la production de résumés/paraphrases (Nenkova et McKeown, 2012 ; Colin et Gardent, 2018 ; Nazari et Mahdavi, 2018), la génération de questions (Zhao *et al.*, 2018) mais aussi la correction orthographique (Grundkiewicz et Junczys-Dowmunt, 2018). Contrairement à l'image dont les représentations sont continues, la nature discrète du langage est la source de la complexité du problème. En particulier, il est nécessaire de maîtriser un vocabulaire de grande taille (parfois plus de 10^5 mots) et d'ainsi faire face à un nombre de combinaisons important. (Yang *et al.*, 2017).

Le domaine de la génération en langage naturel a longtemps été dominé par des méthodes basées sur des règles créées et ajustées manuellement en collaboration avec les experts (Kukich, 1983 ; Oremus, 2014 ; Reiter *et al.*, 2005) jusqu'à l'arrivée récente de méthodes complètement automatisées en traduction automatique (Kalchbrenner et Blunsom, 2013) qui permettent l'obtention de résultats inégalés par les précédentes approches (Sutskever *et al.*, 2014 ; Vaswani *et al.*, 2017). En effet, l'apparition de l'apprentissage profond, et plus particulièrement l'apparition des modèles de bout-en-bout (approches *"end-to-end"*), sur les problématiques *texte-vers-texte* a propulsé le domaine de la génération du langage (Lebret *et al.*, 2016). Alors que les approches standards dans ce domaine considère une uniformité des formats des données d'entrée et de sortie (à savoir le texte), un cas particulier réside dans l'utilisation de données non linguistiques (graphiques, tableaux numériques, bases de connaissance) pour générer du texte. Cette approche, appelée *données-vers-texte* (ou encore *"data-to-text"*) ouvre de nombreuses perspectives et de nouvelles applications de la génération du langage naturel. Par exemple, dans le domaine du journalisme, (Oremus, 2014) explique comment *"Los Angeles Time"* a été le premier journal à publier le compte rendu d'un évènement sismique grâce à un texte généré automatiquement. Également, les approches *"data-to-text"* peuvent être exploitées pour le suivi des flux numériques (bourses, box-office, suivi de population, etc.), l'aide aux diagnostics médicaux ou encore l'accompagnement d'enfants avec des difficultés d'élocution, par exemple pour les aider à mieux retranscrire leurs journées (Black *et al.*, 2010). Cependant, cette tâche reste encore difficile ; bien que les résultats soient prometteurs, (Reiter *et al.*, 2005 ; Kacprzyk et Zadrożny, 2010 ; Gatt *et al.*, 2014) ont démontré que les textes générés requièrent la complémentarité d'expertise humaine pour certaines tâches.

En recherche d'information (RI), la transduction de données non linguistiques vers un texte adéquat semble également prometteuse : (Bing Image Search Relevance Team, 2018) explique comment la description d'image (Vinyals *et al.*, 2017) permet d'améliorer les résultats du moteur de recherche *Bing*. On pourrait imaginer une approche similaire avec des données numériques permettant ainsi d'augmenter la représentation ou l'appariement de documents (e.g., publications scientifiques) pour améliorer l'efficacité de la recherche. Récemment, (Kuzi et Zhai, 2019) ont proposé une nouvelle tâche et un jeu de données associé pour commencer à évaluer les avantages du *"data-to-text"* en RI. Les auteurs appliquent différentes méthodes de RI pour retrouver les tableaux et figures numériques de papiers de recherche à partir de requêtes textuelles. Dans leurs expériences, les descriptions des figures sont extraites du texte qui les entoure. De plus, (Kahou *et al.*, 2017) introduisent *FigureQA*, un jeu de données de questions-réponses entièrement basées sur des données non linguistiques, et proposent plusieurs architectures de modèles comme référence sur la tache.

Ce papier présente un état de l'art autour de la thématique du *"data-to-text"* en se focalisant sur les techniques d'apprentissage profond. Il se situe donc dans la lignée de (Gatt et Krahmer, 2018) qui explore l'ensemble de la NLG, et présente l'intuition et les détails des techniques et modèles récents en *"data-to-text"*. Là où (Gatt et Krahmer, 2018) proposent une approche exhaustive de la génération de langage (historique, ensemble des taches de génération, etc.) nous choisissons d'explorer avec plus de détails uniquement les approches basées sur les réseaux de neurones profonds et de discuter des perspectives de recherche qui nous semblent prometteuses. Plus particulièrement, notre article repose sur les contributions suivantes :

– Nous présentons un état de l'art qui pose la problématique du *"data-to-text"*, présente les principaux modèles d'apprentissage en NLG et se focalise sur les travaux récents spécifiques à la problématique du *"data-to-text"*.

– Nous exposons notre vision des futures perspectives dans le domaine du *"data-to-text"* et proposons quelques pistes de recherche qui seront abordées durant notre thèse.

La suite de cet article se structure comme suit. Après avoir défini la tâche formellement (Section 2) et présenté l'architecture type d'un modèle de traduction dont s'inspire les modèles de génération du langage naturel et par conséquent ceux liés à la problématique *"data-to-text"* (Section 3.1), nous nous focalisons sur les principales contributions liées au *"data-to-text"* (Section 3.2). La Section 4 discute des enjeux qui restent à résoudre, et proposera des pistes de recherche que nous envisageons d'aborder durant ces trois prochaines années. Enfin, la section 5 conclue le papier.

2. Préliminaires

Parmi les nombreux types de données non linguistiques (graphiques, tableaux numériques, base de connaissance), une source de données émergente dans la littérature retient notre attention, à savoir les données numériques présentées sous forme d'un tableau. L'objectif est alors de générer une description (e.g., transcription fidèle, résumé, ou simple explication sur le contenu). Dans ce qui suit, nous définissons les notations de ces tableaux numériques et formalisons la problématique du *"data-to-text"*. Afin de formaliser les notations, nous reprenons les notations introduites dans (Liang *et al.*, 2009) et reprises par (Wiseman *et al.*, 2017).

2.1. *Notations*

Tableau. Un tableau (en anglais, "a set of records.") s est décrit comme l'ensemble des cellules $r_j : s \coloneqq \{r_j\}_{j=1}^{J}$ où J exprime le nombre de cellules du tableau. Pour chaque cellule $r_j \in s$, nous dénotons:

– $r.t$ son type (*i.e.* le nom de la colonne).

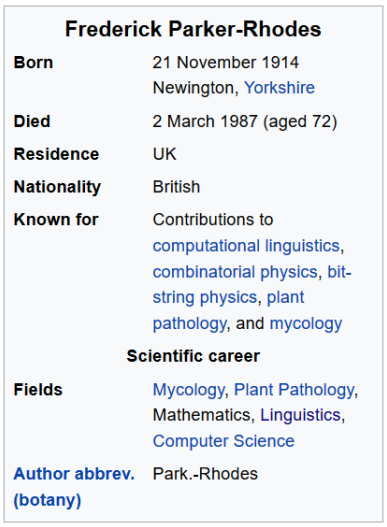
– $r.v$ sa valeur (*i.e.* la valeur de la cellule). La valeur d'une cellule peut contenir plusieurs mots (allant jusqu'à plusieurs dizaines dans le jeu de données WikiBIO (Lebret *et al.*, 2016)). Il est possible de segmenter ces valeurs en unités plus fines, afin de les disposer dans des cellules séparées. Dans ce cas, il est coutume d'enrichir le *type* de l'élément, et d'y inscrire la position du mot au sein de la cellule originelle. On notera p^+ la position du mot en lisant depuis la gauche, et p^- en lisant depuis la droite. Le *type* de l'élément devient donc $r.t_{motsimple} = (r.t,\ p^+,\ p^-)$.

– $r.e$ l'*entité* de l'élément (*i.e.* la ligne à laquelle il appartient lorsque le tableau contient plusieurs lignes).

Par exemple, le tableau 1 décrit les résultats d'un match de basket-ball issu de RotoWire (Wiseman *et al.*, 2017), et dont les lignes regroupent les statistiques par équipe. Un élément du tableau 1 peut être représenté par $(r.t,\ r.v) = (\text{TEAM},\ \text{Heat})$ dans le cadre d'une représentation sans tenir compte des entités, ou $(r.e,\ r.t,\ r.v) = (1,\ \text{TEAM},\ \text{Heat})$ pour prendre en compte la structure par ligne du tableau.

TEAM	WIN	LOSS	POINTS	CITY	...
Heat	11	12	95	Miami	...
Hawks	7	15	103	Atlanta	...

Tableau 1. *Exemple de tableau du jeu de données RotoWire (Wiseman* et al.*, 2017). Le texte à générer à partir de cette exemple commence par : 'The Atlanta Hawks defeated the Miami Heat, 103 - 95 [...]'*



Frederick Parker-Rhodes

Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
	Scientific career
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Author abbrev. (botany)	Park.-Rhodes

La première phrase de l'article Wikipédia associé au tableau est :

"Frederick Parker-Rhodes (21 March 1914 – 21 November 1987) was an English linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist."

Figure 1. *Exemple d'une paire (tableau, description), tirée de la base de données WikiBIO (Lebret* et al.*, 2016).*

Un autre exemple de tableau est présenté dans la Figure 1 où le tableau ne contient qu'une seule entité (ici Frederick Parker-Rhodes). Dans ce cas, chaque élément r du tableau n'est défini que par $r.t$ et $r.v$: par exemple $r = (\text{Residence}, \text{UK})$, où le nom de colonne est $r.t =$ 'Residence' et la valeur de la cellule $r.v =$ 'UK'. De plus, cet exemple est une illustration du fait que des cellules peuvent contenir plusieurs mots. La cellule (Name, Frederick Parker-Rhodes) est découpée en deux nouvelles cellules annotés ainsi : ({Name, 1, 2}, Frederick) et ({Name, 2, 1}, Parker-Rhodes).

2.2. *Formulation du problème*

A partir du tableau $s = \{r_j\}_{j=1}^{J}$, l'objectif est de générer une description adéquate en langage naturel : nous dénotons $\hat{y} = \hat{y}_{1:T} = (\hat{y}_1, \ldots, \hat{y}_T)$ une telle description, où T est le nombre de mots qui la composent. Cette description générée $\hat{y}$ doit être la plus proche possible de l'instance de supervision $y = y_{1:T_s}$ issue du jeu de données associée au tableau s ; où T_s représente la taille du texte associé au tableau s. Un jeu de données regroupe N paires d'exemples $\left\{(s^n, y^n_{1:T_s})\right\}_{n=1}^{N}$. L'objectif d'un modèle *"data-to-text"* est de maximiser la log-vraisemblance de l'ensemble de ces exemples de la façon suivante :

$$\arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \sum_{n=1}^{N} \log P(\hat{y}_{1:T} = y^n_{1:T} \mid s^n; \theta) \qquad [1]$$

où θ représente l'ensemble des paramètres du modèle et $P(\hat{y}^n_{1:T} = y^n_{1:T} \mid s^n; \theta)$ la probabilité de générer la bonne séquence $\hat{y}^n_{1:T}$ pour le tableau s^n. En phase d'inférence, le but est de générer une séquence $\hat{y}^*_{1:T}$ qui maximise, pour un tableau quelconque s en utilisant la formule des probabilités totales :

$$\hat{y}^*_{1:T} = \arg\max_{\hat{y}_{1:T}} \prod_{t=1}^{T} P(\hat{y}_t | \hat{y}_{1:t-1}, s) \quad [2]$$

où la probabilité a posteriori $P(\hat{y}_t|\hat{y}_{1:t-1}, s)$ exprime la probabilité de générer le terme $\hat{y}_t$ pour le tableau s étant donnée les $t-1^{\text{ième}}$ termes $\hat{y}_{1:t-1}$ générés précédemment.

3. État de l'art

Cette section présente dans un premier temps les architectures les plus influentes en traduction machine basée sur l'apprentissage profond, à savoir les approches de type encodeur-décodeur. Il est important de noter que d'autres approches existent telle que la modélisation d'alignements, qui cherche à apprendre statistiquement des alignements entre des mots ou groupes de mots (Och *et al.*, 1999 ; Koehn *et al.*, 2003). Ensuite, nous analysons les modèles qui contribuent à la problématique *"data-to-text"*.

3.1. *Approches génératives de type encodeur-décodeur*

A ce jour, les approches de la traduction par apprentissage profond sont basées sur les modèles *encodeur-décodeur* (Sutskever *et al.*, 2014 ; Cho *et al.*, 2014). Ces architectures, également appelées séquence-vers-séquence (où encore seq2seq) passent obligatoirement par une étape d'encodage du tableau d'entrée afin de construire sa représentation latente (*embedding*) : il s'agit d'apprendre une représentation vectorielle dense du vocabulaire ou des données dans un espace de petite dimension (généralement 10^2). Dans ce qui suit, nous décrivons les principales architectures de type encodeur-décodeur. Nous présentons ces approches en considérant que la donnée en entrée est une séquence de mots, hypothèse faite par la plupart des approches de génération du langage. Les architectures consacrées aux données non-linguistiques seront présentées dans la sous-section suivante.

Réseaux récurrents. Les réseaux récurrents (*Recurrent Neural Networks* ou RNN) permettent d'appréhender les séquences de termes tout en évitant les représentations à faible densité et l'explosion du nombre de paramètres. Les réseaux récurrents possèdent un vecteur d'état h_t mis à jour par chaque nouveau mot y_t de la séquence en tenant compte de l'état précédent h_{t-1} : $h_t = rnn(y_t, h_{t-1})$. Afin de modéliser les interactions avec les mots futurs,

il est aussi possible de parcourir la séquence dans le sens inverse de la lecture. Dans ce cas, $h_t = [h_t^+; h_t^-]$ est la concaténation des vecteurs d'état et le réseau est appelé réseau récurent bidirectionnel, ou bi-RNN. Les séquences similaires partagent ainsi une représentation dense et de petite dimension. Dans leur grand travail exploratoire, (Britz *et al.*, 2017) ont empiriquement démontré que les réseaux récurrents à mémoire (*Long Short-Term Memory networks* (LSTMs)) (Hochreiter et Schmidhuber, 1997) sont plus performants que les autres variantes de RNNs.

Les architectures *encodeur-décodeur* utilisent généralement deux RNNs : un premier qui sert à encoder une séquence de taille variable en un vecteur de dimension fixe appelé *contexte* (usuellement le dernier vecteur d'état, h_T) et un deuxième qui sert de générateur et "décode" le contexte mot par mot :

$$p(y_t | \{y_1, ..., y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, s_t, c) \quad [3]$$

$$s_t = rnn(s_{t-1}, y_{t-1}, c) \quad [4]$$

où $\mathbf{x}$ représente l'entrée encodée, c le contexte, y_k la représentation latente du $k^{\text{ième}}$ élément du tableau. s_t et s_{t-1} dénotent les vecteurs d'état du RNN décodeur aux pas de temps t et $t-1$ respectivement.

Mécanisme d'attention. L'utilisation d'un vecteur de taille fixe (et donc limité dans la quantité d'information qu'il peut retenir) rend complexe l'encodage de longues séquences. Une réponse à cet enjeu réside dans l'introduction du mécanisme d'attention par (Bahdanau *et al.*, 2014), qui permet au décodeur de sélectionner à chaque pas de temps un passage de l'entrée sur lequel se concentrer pour générer le mot suivant. Dans les faits, le décodeur apprend itérativement à pondérer les vecteurs d'état de l'encodeur, et d'avoir ainsi un contexte variable à chaque pas de temps de la génération. Dans les équations 3 et 4, le contexte c_t à l'étape t est estimé par moyenne pondérée des états appris au cours de la séquence $\mathbf{x}$:

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \quad [5]$$

$$\text{avec } \alpha_{tj} = a(h_j, y_{t-1}) \quad [6]$$

Où les α_{tj} sont eux-mêmes le resultat d'une fonction de correspondance ($a(.,.)$) entre chaque vecteur d'état h_j et y_{t-1}. En pratique, le coefficient d'importance α_{tj} du mot x_t est normalisé grâce à la fonction softmax.

Mécanisme de Copie. Un problème majeur de la génération de texte vient de la difficulté de gérer (et générer) les mots rares ou jamais vus en entraînement (Nom propre, acronyme, etc.). Pour qu'un réseau standard lise puis prédise un mot rare, il devra encoder le mot et le garder dans son vecteur d'état pendant de nombreux pas de temps. Comme le vecteur d'état a une capacité de stockage limitée et que l'optimisation de tels modèles est sujette aux problèmes de disparition de gradient, c'est une opération complexe,

voire impossible pour les mots jamais vus jusqu'à présent. Bien que les mécanismes d'attention introduits plus haut permettent de faire face à une partie du problème, le mécanisme classique de softmax ne permet pas de générer des mots rares ou inconnus jusqu'ici (Sutskever *et al.*, 2014). Récemment, (Vinyals *et al.*, 2015) ont introduit les réseaux pointeurs qui génèrent des éléments inconnus : en permettant la copie d'un mot de la phrase d'entrée au travers des poids d'attention qui servent de pointeurs à chaque pas de temps. Cependant, ces réseaux ne peuvent pas générer de mots qui n'existent pas dans la phrase d'entrée et ne sont donc pas adaptés à la génération de texte. Pour remédier à ce problème (Gulcehre *et al.*, 2016 ; Merity *et al.*, 2016 ; See *et al.*, 2017) proposent des architectures hybride qui alternent entre copie d'un mot présent en entrée et génération d'un mot nouveau. Pour ce faire, les réseaux sont munis d'une sentinelle $p_t^{\text{gen}} = s(y_{t-1}, s_t, c_t, p_{t-1}^{\text{gen}})$ qui se charge de décider de quelle distribution tirer le mot du pas de temps t en fonction de s une fonction différentiable quelconque, y_{t-1} l'embedding du mot précédemment généré, s_t le vecteur d'état du RNN décodeur, c_t le context appris par l'encodeur pour le pas de temps t et p_{t-1} la valeur de la sentinelle au pas de temps $t-1$. Si l'on note $\mathcal{D}_{\text{vocab}}$ la distribution apprise par le réseau sur l'ensemble des mots connus et $\mathcal{D}_{\text{att}}$ la distribution d'attention apprise sur les éléments en entrée, alors la distribution finale est donnée par :

$$\mathcal{D} = p^{\text{gen}}\mathcal{D}_{\text{vocab}} + (1 - p^{\text{gen}})\mathcal{D}_{\text{att}} \quad [7]$$

3.2. *Vers des modèles génératifs pour le "data-to-text"*

Nous nous focalisons dans cette section sur les modèles les plus récents de la littérature NLG qui s'intéressent aux données non linguistiques. Ces modèles cherchent à simplement transcrire le contenu du tableau en texte en se basant sur des modèles de traduction. Rappelons qu'un tableau est ici vu comme une séquence de mots labélisés par le nom de la colonne dans laquelle il apparaît (et plus rarement l'index de la ligne) : $s = \{(r_j.t,\ r_j.v)\}_{j=1}^{J}$. Toutes les architectures décident d'abord de calculer séparément l'embedding du *type* $\mathbf{r}.t$ et celui de la *valeur* $\mathbf{r}.v$ pour ensuite appliquer une fonction non linéaire à cette paire et obtenir un embedding pour la cellule : $\mathbf{s} = \{\mathbf{r}_j\}_{j=1}^{J}$.

Dans un premier temps, (Liu *et al.*, 2017) décident d'intégrer l'information structurelle des tableaux directement au travers des différentes briques qui composent leur réseau. Les auteurs proposent une version modifiée du LSTM qui prend pour entrée à la fois l'embedding de la cellule ($\mathbf{r}_j$ selon les notations du paragrapge précédent) et l'embedding du *type* de la cellule ($\mathbf{r}_j.t$). Le mécanisme d'attention aussi est impacté. Les auteurs proposent une attention hybride, calculée à partir de deux vecteurs d'attention intermédiaires : un premier sur l'ensemble des cellules $\{\mathbf{r}_j\}_{j=1}^{J}$ et un second uniquement sur leur *type* $\{\mathbf{r}_j.t\}_{j=1}^{J}$. Le vecteur d'attention final est calculé comme le produit terme-à-

terme de ces deux vecteurs intermédiaires. De nombreuses autres approches suivent la même idée, telle que par exemple (Sha *et al.*, 2017 ; Wang, 2019) qui apportent chacun une pondération différente pour les vecteurs intermédiaires d'attention. Par soucis de concision, nous ne les détaillons pas. Ces approches obtiennent des résultats très encourageants pour la génération de séquences de petite taille, comme par exemple une phrase, mais sont sérieusement limitées pour des générations de plus grande taille, comme un paragraphe (voir (Wiseman *et al.*, 2017)). Elles ont aussi le désavantage de ne pas être contrôlables ou interprétables.

Pour rendre ces approches plus modulables et interprétables, (Puduppully *et al.*, 2018) proposent de renouer avec les approches plus anciennes qui séparaient explicitement la planification (*quoi dire*) de la réalisation (*comment le dire*). Les auteurs utilisent un premier décodeur qui reprend l'idée des réseaux pointeurs afin de générer une suite d'élément du tableau qui servira de plan. Un deuxième réseau récurrent a ensuite pour objectif de générer la description, conditionnellement au plan. Le plan "parfait" est donné par l'hypothèse que tout mot figurant à la fois dans le tableau d'entrée et sa description est une entité qui fait partie du plan. Le réseau est ensuite entraîné pour minimiser la distance entre le plan généré et le vrai plan ainsi que la distance entre la séquence générée et la vraie séquence. Cette méthode a l'avantage de permettre la génération de textes plus longs, en donnant au deuxième décodeur l'occasion de se concentrer sur l'écriture de phrases valides pour relier deux entités du plan. Le plan donne aussi une première interprétation des séquences générées. Bien que les auteurs ne le mentionnent pas, un système ainsi entraîné pourrait générer un texte dont le plan lui aura été fourni par un utilisateur. C'est l'approche choisie par (Wiseman *et al.*, 2018) qui mentionnent explicitement l'interaction utilisateur-algorithme et proposent d'intégrer explicitement la modélisation d'un plan dans l'architecture du réseau. Plutôt que d'utiliser un seul RNN pour générer la séquence dans son intégrité, ils utilisent un Modèle de Semi-Markov Caché (HSMM) qui coordonne différents RNNs : le modèle apprend une représentation et une matrice de transitions entre plusieurs états. A chaque pas de temps, le décodeur transite vers un état et décide d'un nombre de mots à générer : ce sera le RNN associé à cet état qui fera la génération de cette sous-séquence. La suite des états peut être vue comme une *template*, autrement dit un plan. Par exemple, pour un restaurant, le plan [nom, localisation, prix] indique au réseau de mentionner d'abord le nom, ensuite l'endroit puis finalement le prix du restaurant. Contrairement à (Puduppully *et al.*, 2018), les transitions et les états cachés sont appris en toute autonomie par le réseau et aucune hypothèse a priori n'est faite. Cette méthode permet aussi, après l'entraînement de spécifier le plan et de générer un texte selon un schéma prédéfini par l'utilisateur (afin par exemple de censurer un prix, ou une information personnelle comme un numéro de téléphone dans un agent conversationnel).

Au delà des architectures précises de chaque modèle, les performances des systèmes de NLG dépendent en grandes partie de la qualité des jeux de données

d'apprentissage. Pour l'instant la communauté dispose de plusieurs jeux de données dont les plus utilisés sont :

– **WikiBIO** (Lebret *et al.*, 2016), constitué de $700K+$ instances tirées de biographies rédigées sur Wikipédia[1]. L'enjeu consiste à rédiger les premières phrases d'un article en utilisant l'*infobox* (voir exemple 1).

– **E2E** (Novikova *et al.*, 2017), où les auteurs proposent des descriptions tabulaires de restaurant accompagnées de leurs transcriptions en texte.

– **RotoWire** (Wiseman *et al.*, 2017), qui regroupe plus de 3000 résultats de matchs de basket-ball et de l'article de journal qui les accompagne (voir exemple du Tableau 1).

On trouve d'autres jeux de données moins utilisés récemment dans (Perez-Beltrachini et Gardent, 2017), où les auteurs proposent une méthodologie rigoureuse pour comparer les différences entre ces jeux de données (en comparant le nombre d'exemple, mais aussi la complexité des phrases à générer, l'adéquation entre l'entrée et la sortie attendue, etc.).

4. Perspectives de thèse

Malgré les importants progrès que l'influence de la traduction a pu avoir sur le domaine du *"data-to-text"*, rappelons que les modèles passés en revue encodent les tableaux de façon 'simple', sans transformation/croisement de données. A notre avis, il reste des spécificités pour l'instant peu ou pas abordées qui méritent l'attention de la communauté, comme la capacité des systèmes de NLG à faire de l'inférence. Un autre aspect qui nous semble important réside dans l'évaluation de ces systèmes avec la définition de métriques et de tâches permettant de mesurer leur performance. Nous présentons dans ce qui suit les différentes pistes de recherche que nous envisageons.

Vers l'inférence comparative. La traduction consiste à générer un texte basé en majeure partie sur le texte d'entrée. La plupart des travaux (Lebret *et al.*, 2016 ; Liu *et al.*, 2017 ; Wiseman *et al.*, 2018) se concentre sur une description d'une donnée non linguistique basée sur l'énumération de ses caractéristiques. Nous pensons que la prochaine étape est de dépasser ce paradigme de simple description et de permettre aux systèmes de réaliser des inférences/croisements entre les données. Un système de NLG performant doit donc être capable de (1) sélectionner uniquement les données pertinentes et (2) inférer de nouvelles données qui ne sont pas explicitement contenues en entrée. Par exemple, dans le jeu de données RotoWire (Wiseman *et al.*, 2017) (Table 1), les statistiques des parties de basket-ball enregistrées sont pour certaines inutiles et pour d'autres incomplètes. Il peut être intéressant, par exemple d'identifier l'équipe gagnante et donc d'inférer cette information à partir des données du

1. www.wikipedia.org

tableau. (Nie *et al.*, 2018) proposent un premier pas dans cette direction en enrichissant manuellement les tableaux d'entrées de toutes les opérations connues, et en introduisant un mécanisme d'attention supplémentaire pour parcourir ces nouvelles données. Malheureusement, une telle pratique a de nombreuses lacunes : elle est trop coûteuse pour être applicable et peut rapidement devenir demandeuse de ressources pour effectuer l'ensemble des calculs possibles. Dans leurs travaux, ils ne procèdent qu'à deux opérations à la fois (les différences deux à deux des cellules à valeurs numériques et l'opérateur argmax sur les colonnes à valeurs numériques) ; il devient difficile de proposer des opérations plus complexes (qui requièrent par exemple quatre arguments) sans augmenter significativement le coût.

Nous proposons donc de nous intéresser à cette problématique d'inférence en réfléchissant à la formalisation de modèles neuronaux, proche des mécanismes de copie et d'attention. Ces derniers pourraient proposer/identifier une opération (parmi une liste spécifiée avant l'entraînement) et une séquence d'arguments à laquelle appliquer l'opération. Cette dernière serait appliquée aux arguments à la volée, indépendemment du réseau de neurones. Par exemple, l'opération *différence absolue* pourrait être appliquée entre n'importe quelle paire de cellule contenant des valeurs numériques. Le module apprendrait donc à sélectionner la paire du tableau la plus pertinente sur laquelle calculer la différence en valeur absolue, et à utiliser le résultat pour générer le prochain mot. Un tel module devra être entièrement différentiable. Mentionnons (Trask *et al.*, 2018) qui introduisent un nouveau module capable d'opérations arithmétiques simples, qui apprend par exemple à compter les pas de temps. Nos problématiques imposent des opérations complexes (groupement de lignes suivant la valeur d'une colonne par exemple) et très variées, nous chercherons donc une adaptation du module de (Trask *et al.*, 2018) vers de telles opérations.

Vers des tâches-métriques pour l'évaluation des modèles. Jusqu'à présent, l'évaluation des systèmes de NLG passe par la métrique BLEU (Papineni *et al.*, 2002). Cette métrique base son score sur la comparaison des n-grams contenus dans la phrase proposée par le système et la référence humaine. Cette métrique convient plutôt bien aux systèmes de traduction étant donnée qu'il existe peu de manières très différentes de dire exactement la même chose (à l'alignement prêt : (Banerjee et Lavie, 2005) propose un alignement des deux phrases avant d'appliquer le score BLEU). Cette hypothèse se révèle très vite fausse dans le cadre général de la NLG : deux résumés n'ont aucune raison d'avoir un nombre important de mots en commun, si ce n'est les entités du passage d'entrée. Pour tirer partie de ce fait, (Wiseman *et al.*, 2017) proposent une nouvelle métrique sous la forme d'une nouvelle tâche : les auteurs entraînent un réseau à correctement labéliser les paires (*entité*, *valeur*) par leur *type*, étant donné la phrase dans laquelle elles se trouvent. Par exemple, dans la phrase du tableau 1 *The Atlanta Hawks defeated the Miami Heat, 103 - 95 [...]*, la paire (1, 103) doit être labélisée par POINTS (dans cette exemple l'entité est *Atlanta Hawks* qui est représenté par le numéro de sa ligne dans le

tableau). Un tel modèle entraîné pour répondre à cette tâche sur les exemples d'entraînement du jeu de données obtient un score proche de la perfection sur une partie non vue à l'entraînement. Couplé à un modèle de langue (Krause *et al.*, 2016) qui juge de la qualité grammaticale des textes générés, ce réseau peut servir de métrique neuronale et pourrait remplacer BLEU dans le cadre de génération libre : un texte proposé par un modèle sera noté en fonction de la facilité avec laquelle la métrique neuronale labélise les paires d'entités qui s'y trouvent, ainsi que de la fluidité que lui accorde le modèle de langue.

De notre point de vue, il semble intéressant d'identifier des tâches intermédiaires permettant l'évaluation des modèles *"data-to-text"* basés sur des inférences complexes, et dont l'évaluation automatique d'un résumé semble une tâche compliquée et incertaine. Par exemple, nous pouvons nous inspirer de (Kahou *et al.*, 2017) qui introduisent une nouvelle tâche, *figureQA*, dont le but est de répondre à des questions parfois complexes, associées à des figures type diagrammes ou graphes (e.g., 'Est-ce que X a la plus grande valeur ?'). Nous proposons la création d'une tâche similaire de type question-réponse basée sur des tableaux où les description des tableaux pourraient être un intermédiaire pour répondre aux questions. Si les descriptions générées par un modèle permettent de répondre aux questions aussi bien que l'objet initial, la tâche de question-réponse devient ainsi une tâche-métrique pour l'évaluation de la qualité de la description. Nous pensons ainsi construire ce jeu de données permettant un nouveau cadre d'évaluation au travers d'une tâche-métrique à partir de jugements humains (*"crowdsourcing"*) et rendre par la suite ce jeu de données accessible à la communauté.

5. Conclusion

Dans ce papier, nous avons présenté l'état de l'art des systèmes de génération de langage à partir de données non linguistiques. Nous avons ensuite proposé plusieurs pistes de recherche dans ce domaine. Premièrement, nous avons aussi émis l'idée d'un nouveau module neuronal qui serait capable d'enrichir à la volée un tableau au moyen d'inférences. Finalement, nous proposons également des éléments de réflexion sur l'adoption de nouvelles métriques, afin de suppléer à la métrique BLEU (et aux métriques associées) qui nous semble présenter de sérieuses lacunes hors du domaine de la traduction. Pour ce faire, nous pensons nécessaire d'introduire de nouveaux jeux de données adaptés à ces nouvelles problématiques ainsi qu'au domaine particulier de la RI.

6. Bibliographie

Bahdanau D., Cho K., Bengio Y., « Neural Machine Translation by Jointly Learning to Align and Translate », *arXiv:1409.0473 [cs, stat]*, September, 2014. arXiv: 1409.0473.

Banerjee S., Lavie A., « METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments », *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Association for Computational Linguistics, Ann Arbor, Michigan, p. 65-72, June, 2005.

Bing Image Search Relevance Team, « Internet-Scale Deep Learning for Bing Image Search », 2018.

Black R., Reddington J., Reiter E., Tintarev N., Waller A., « Using NLG and Sensors to Support Personal Narrative for Children with Complex Communication Needs », *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*, Association for Computational Linguistics, Los Angeles, California, p. 1-9, June, 2010.

Britz D., Goldie A., Luong M.-T., Le Q., « Massive Exploration of Neural Machine Translation Architectures », *arXiv:1703.03906 [cs]*, March, 2017. arXiv: 1703.03906.

Cho K., van Merrienboer B., Bahdanau D., Bengio Y., « On the Properties of Neural Machine Translation: Encoder-Decoder Approaches », *arXiv:1409.1259 [cs, stat]*, September, 2014. arXiv: 1409.1259.

Colin E., Gardent C., « Generating Syntactic Paraphrases », p. 7, 2018.

Gatt A., Krahmer E., « Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation », *Journal of Artificial Intelligence Research*, vol. 61, p. 65-170, January, 2018.

Gatt A., Portet F., Reiter E., Hunter J., Mahamood S., Moncur W., Sripada S., « From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management », p. 34, 2014.

Grundkiewicz R., Junczys-Dowmunt M., « Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation », *arXiv:1804.05945 [cs]*, April, 2018. arXiv: 1804.05945.

Gulcehre C., Ahn S., Nallapati R., Zhou B., Bengio Y., « Pointing the Unknown Words », *arXiv:1603.08148 [cs]*, March, 2016. arXiv: 1603.08148.

Hochreiter S., Schmidhuber J., « LSTM can solve hard log time lag problems », p. 8, 1997.

Kacprzyk J., Zadrożny S., « Towards human consistent data driven decision support systems using verbalization of data mining results via linguistic data summaries », *Bulletin of the Polish Academy of Sciences: Technical Sciences*, January, 2010.

Kahou S. E., Michalski V., Atkinson A., Kadar A., Trischler A., Bengio Y., « FigureQA: An Annotated Figure Dataset for Visual Reasoning », *arXiv:1710.07300 [cs]*, October, 2017. arXiv: 1710.07300.

Kalchbrenner N., Blunsom P., « Recurrent Continuous Translation Models », p. 10, 2013.

Koehn P., Och F. J., Marcu D., « Statistical Phrase-Based Translation », p. 7, 2003.

Krause B., Lu L., Murray I., Renals S., « Multiplicative LSTM for sequence modelling », *arXiv:1609.07959 [cs, stat]*, September, 2016. arXiv: 1609.07959.

Kukich K., « Design of a knowledge-based report generator », Association for Computational Linguistics, p. 145, 1983.

Kuzi S., Zhai C., « Figure Retrieval from Collections of Research Articles.pdf », 2019.

Lebret R., Grangier D., Auli M., « Neural Text Generation from Structured Data with Application to the Biography Domain », *arXiv:1603.07771 [cs]*, March, 2016. arXiv: 1603.07771.

Liang P., Jordan M. I., Klein D., « Learning semantic correspondences with less supervision », vol. 1, Association for Computational Linguistics, p. 91, 2009.

Liu T., Wang K., Sha L., Chang B., Sui Z., « Table-to-text Generation by Structure-aware Seq2seq Learning », *arXiv:1711.09724 [cs]*, November, 2017. arXiv: 1711.09724.

Merity S., Xiong C., Bradbury J., Socher R., « Pointer Sentinel Mixture Models », September, 2016.

Nazari N., Mahdavi M. A., « A survey on Automatic Text Summarization », p. 16, 2018.

Nenkova A., McKeown K., « A Survey of Text Summarization Techniques », *in* C. C. Aggarwal, C. Zhai (eds), *Mining Text Data*, Springer US, Boston, MA, p. 43-76, 2012.

Nie F., Wang J., Yao J.-G., Pan R., Lin C.-Y., « Operations Guided Neural Networks for High Fidelity Data-To-Text Generation », *arXiv:1809.02735 [cs]*, September, 2018. arXiv: 1809.02735.

Novikova J., Dušek O., Rieser V., « The E2E Dataset: New Challenges For End-to-End Generation », *arXiv:1706.09254 [cs]*, June, 2017. arXiv: 1706.09254.

Och F. J., Tillmann C., Ney H., « Improved Alignment Models for Statistical Machine Translation », p. 9, 1999.

Oremus W., « The First News Report on the L.A. Earthquake Was Written by a Robot », March, 2014.

Papineni K., Roukos S., Ward T., Zhu W.-J., « Bleu: a Method for Automatic Evaluation of Machine Translation », *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, p. 311-318, July, 2002.

Perez-Beltrachini L., Gardent C., « Analysing Data-To-Text Generation Benchmarks », Association for Computational Linguistics, p. 238-242, 2017.

Puduppully R., Dong L., Lapata M., « Data-to-Text Generation with Content Selection and Planning », *arXiv:1809.00582 [cs]*, September, 2018. arXiv: 1809.00582.

Reiter E., Dale R., *Building natural language generation systems*, Studies in natural language processing, Cambridge Univ. Press, Cambridge, 2000. OCLC: 246426783.

Reiter E., Sripada S., Hunter J., Yu J., Davy I., « Choosing words in computer-generated weather forecasts », *Artificial Intelligence*, vol. 167, nº 1-2, p. 137-169, September, 2005.

See A., Liu P. J., Manning C. D., « Get To The Point: Summarization with Pointer-Generator Networks », *arXiv:1704.04368 [cs]*, April, 2017. arXiv: 1704.04368.

Sha L., Mou L., Liu T., Poupart P., Li S., Chang B., Sui Z., « Order-Planning Neural Text Generation From Structured Data », *arXiv:1709.00155 [cs]*, September, 2017. arXiv: 1709.00155.

Sutskever I., Vinyals O., Le Q. V., « Sequence to Sequence Learning with Neural Networks », p. 9, 2014.

Trask A., Hill F., Reed S., Rae J., Dyer C., Blunsom P., « Neural Arithmetic Logic Units », *arXiv:1808.00508 [cs]*, August, 2018. arXiv: 1808.00508.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser [U+FFFD], Polosukhin I., « Attention is All you Need », p. 11, 2017.

Vinyals O., Fortunato M., Jaitly N., « Pointer Networks », p. 9, 2015.

Vinyals O., Toshev A., Bengio S., Erhan D., « Show and Tell: Lessons Learned from the 2015MSCOCO Image Captioning Challenge », 2017.

Wang Y., « KG-to-Text Generation with Slot-Aware Attention and Position Self-Attention », 2019.

Wiseman S., Shieber S. M., Rush A. M., « Challenges in Data-to-Document Generation », *arXiv:1707.08052 [cs]*, July, 2017. arXiv: 1707.08052.

Wiseman S., Shieber S., Rush A., « Learning Neural Templates for Text Generation », p. 14, 2018.

Yang Z., Dai Z., Salakhutdinov R., Cohen W. W., « Breaking the Softmax Bottleneck: A High-Rank RNN Language Model », *arXiv:1711.03953 [cs]*, November, 2017. arXiv: 1711.03953.

Zhao Y., Ni X., Ding Y., Ke Q., « Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks », p. 10, 2018.