Search-based and supervised text generation

Betty Fabre

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Betty FABRE

Search-Based and Supervised Text Generation

Thèse présentée et soutenue à Lannion, le 16 septembre 2022
Unité de recherche : IRISA, UMR CNRS 6074

Rapporteurs avant soutenance :

John KELLEHER	Professeur des Universités, TU Dublin
Stefan RIEZLER	Professeur des Universités, Heidelberg University

Composition du Jury :

Présidente :	Pascale SEBILLOT	Professeure des Universités, IRISA
Examinateurs :	Christophe CERISARA	Professeur des Universités, LORIA
	Pascale SEBILLOT	Professeure des Universités, IRISA
	François YVON	Professeur des Universités, LISN
Dir. de thèse :	Damien LOLIVE	Maître de confrences, IRISA
Co-dir. de thèse :	Jonathan CHEVELU	Maître de confrences, IRISA
	Tanguy URVOY	Chercheur, Orange Labs Lannion

Le petit chat se promène.

Unknown

La fin du désert se cache peut-être derrière chaque dune.

Orelsan

Acknowledgement

After these 3 years (+ 6 months) working on this PhD, I would like to thank all those who made this work possible.
First, I want to thank John Kelleher and Stefan Riezler for accepting to review this dissertation. I would also like to thank Pascale Sebillot, Christophe Cerisara and François Yvon for accepting to be the members of the jury.

I also wish like to thank the members of my CSID, Artem and Pierre-François for their feedback on my work throughout this PhD.

I was very lucky to be able to do my research at Orange Labs in Lannion. I would like to thank you, Tanguy, for sharing your office with me where I learned a lot. Thank you for your constant support at each step of this PhD. It has been unvaluable and I'm very grateful.

I would also like to thank my academic supervisors Jonathan and Damien without whom this thesis would not have happened. You provided me with excellent working conditions at ENSSAT and your additional guidance and shared insights were very useful.

More generally I would like to thank the PROF (+ corridor) team and the Expression team for the coffee breaks and lunches of course but also for the various discussions and very constructive feedback on my work. This journey would not have been the same without you.

I also want to thank my fellow PhD students Antoine, Meysam, Aghilas, Somayeh, Lily, Valentina, Sébastien, Thibault, Xihui, Pierre, Tangui, Mina and Victor (I hope not to forget anyone). I really enjoyed working or just interacting with you all every day.

On a more personal note, I thank my friends for being so patient. I did not ghost you, I was just finishing writing this manuscript. I thank my family for always asking about my thesis even though I suspect they didn't really understand what I was doing. I especially thank my brother Flavien, for sharing with me his expertise in the Pythagorean theorem.

Finally I would like to thank Melina. I promise from now on I won't say "I'll do this when my thesis is done" anymore.

Abstract

Résumé

Les modèles supervisés **encodeurs-decodeurs** nécessitent de grands jeux de données alignés pour être entraînés. Les données nécessaires ne sont pas encore disponibles pour plusieurs tâches telles que la **verbalisation de triplets RDF** ou la **génération de paraphrases**. D'abord, nous avons exploré la tâche de verbalisation de triplets RDF. Nous avons entraîné des modèles **TRANSFORMER** sur une nouvelle version des données **WebNLG** et avons étudié plusieurs stratégies de pré-entraînement pour surmonter la petite taille du corpus. Ensuite, nous avons étudié la tâche de génération de paraphrases. Nous avons entraîné des modèles TRANSFORMER sur des corpus alignés afin de les comparer directement avec les modèles de la littérature. Une contribution importante de la thèse a été de proposer un **cadre expérimental uniforme** pour comparer les modèles encodeurs-décodeurs pour la génération de paraphrases. Nous avons également suivi la voie des méthodes alternatives basées recherche pour générer des paraphrases. Pour ce faire, nous avons transformé la tâche de génération de paraphrases en un problème de **recherche dans un arbre**. Nous avons ensuite développé deux stratégies de recherche: **MCPG** et **PTS** et un module de score des paraphrases qui exploite le **BERTscore**, **GPT-2** et la **distance de Levenshtein**. Enfin, nous avons mené des expériences de **distillation** avec le modèle TRANSFORMER.

Mots-cléfs : encodeur-decodeur, triplets RDF, paraphrase, recherche dans un arbre, distillation

Abstract

In this thesis, we studied the topic of *Search-Based and Supervised Text Generation*. Supervised **encoder-decoder** models require huge aligned dataset to be trained. The necessary data is not yet available for several tasks such as **RDF triples verbalization** or **paraphrase generation**. First, we explored the data-to-text task of RDF verbalization. We trained supervised **TRANSFORMER models** on a newly released version of the **WebNLG** dataset and studied in depth several pre-training strategies to overcome the small size of the aligned corpus. Then, we studied the paraphrase generation task. We have trained TRANSFORMER models on aligned corpora to directly compare with the literature models. An important contribution of the thesis was to propose a **uniform experimental framework** for comparing encoder-decoder models for paraphrase generation. We also followed the path of search-based alternative strategies to generate paraphrases. The main motivation was to provide a better control of the generated paraphrase. To do so, we casted the paraphrase generation tasks as a **tree-search** problem. We then developped two search strategies **MCPG** and **PTS** and a paraphrase scoring module that leverages the **BERTscore**, **GPT-2** and the **Levenshtein distance**. Finally, we conducted experiments of data **distillation** for the TRANSFORMER model.

Keywords : encoder-decoder, RDF triples, paraphrase , TRANSFORMER, tree-search, distillation

Table of Contents

List of Figures

List of Tables

Synthèse en Français

1 Introduction

Au cours de la dernière décennie, l'apprentissage profond et les réseaux neuronaux ont offert de nouveaux outils qui ont permis l'amélioration de l'état de l'art dans de nombreuses tâches de génération de texte, comme la traduction automatique. Les récents gains de performance en matière de génération de texte ont même fait l'objet d'articles dans les médias généralistes qui les ont qualifiés de *"système d'intelligence artificielle révolutionnaire"*. [1].

À l'instar de ce qui s'est passé dans le domaine de la vision par ordinateur avec les couches de convolution, le domaine du traitement automatique des langues (TALN) a vu le début d'une nouvelle ère avec le développement des réseaux neuronaux. En particulier, trois étapes importantes ont été franchies.

Tout d'abord, les modèles de vectorisation de mots (ou plongement de mots) tels que *word2vec* [Tomas Mikolov, 2012] et *GloVe* [Pennington et al., 2014] remplacent les encodages disjonctifs et sacs de mots, alors largement utilisés, par des représentations vectorielles des mots. Ces représentations sont générées par des réseaux neuronaux qui apprennent un espace où la distance entre les mots est liée à leur similarité sémantique.

Ensuite, les modèles encodeurs-décodeurs du *Seq2Seq* [Cho et al., 2014 ; Sutskever et al., 2014] au *Transformer* [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez et al., 2017a] sont des architectures de bout en bout qui peuvent générer du texte à partir d'une séquence en entrée. Par exemple, pour la traduction automatique, un modèle encodeur-décodeur reçoit une phrase dans une langue et génère sa traduction dans une autre langue. Ces modèles ont permis d'améliorer considérablement les performances sur de nombreuses tâches de génération de texte.

Enfin, très récemment – en même temps que les travaux présentés dans cette thèse –, le domaine du TALN a connu un changement de paradigme majeur avec la sortie d'énormes modèles de langage pré-entraînés Devlin et al. [2018], Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Veselin Stoyanov et al. [2020] et Radford et al. [2019], qui peuvent être spécialisés pour de nombreuses tâches de génération de texte.

Dans cette thèse, nous nous sommes concentrés sur le sujet de **la génération de texte basée recherche et supervisée**. Les modèles supervisés contiennent de plus en plus de paramètres et nécessitent donc de plus en plus de données alignées. Les données nécessaires ne sont pas encore disponibles pour plusieurs tâches telles que la verbalisation de triplets RDF ou la génération de paraphrases. Nous avons

1. "New AI fake text generator may be too dangerous to release, say creators", Alex Hern, The Guardian, 14/02/2019, https://www.theguardian.com/technology/2019/feb/14/elon-musk-backed-ai-writes-convincing-news-fiction

exploré ces deux tâches. Par ailleurs, les réseaux neuronaux sont efficaces mais imparfaits. La course à la performance soulève des questions sur l'évaluation et en particulier sur les métriques d'évaluation automatique utilisées pour évaluer les modèles supervisés. Nous avons rencontré des obstacles dans la reproductibilité de certains modèles – et donc des résultats – publiés dans la littérature.

Nous avons donc étudié les modèles supervisés de génération de texte et avons également exploré des méthodes alternatives basées recherche.

Ce manuscrit est divisé en deux parties.

Dans la partie 4, on donne d'abord une introduction au modèle encodeur-décodeur qui est au cœur de notre travail. Il s'agit donc d'un prérequis essentiel à la lecture des travaux présentés dans ce manuscrit. Dans cette partie, on présente également le travail sur la verbalisation des triplets RDF pour le Challenge WebNLG : une première expérience dans la génération de texte. En effet, un bon moyen d'explorer les dernières avancées en matière de modèles supervisés est de les appliquer à une tâche particulière. C'est ce que nous avons fait en participant au WebNLG Challenge. Le WebNLG Challenge [Gardent, Shimorina et al., 2017b] vise à promouvoir la verbalisation des triplets RDFs. En effet, la précédente édition du challenge ayant eu lieu en 2017, les modèles précédemment soumis au challenge ne s'appuyaient pas sur des Transofrmers et encore moins sur des stratégies de pré-entraînement à grande échelle. Pour le challenge WebNLG 2020, nous avons soumis des Transformers entraînés à l'aide de stratégies d'augmentation de données et de pré-entraînement. Nous rapportons les expériences dans la section 2. Il faut mentionner que le travail sur le challenge WebNLG a été réalisé en collaboration et à contribution égale avec un autre doctorant, Sébastien Montella. Ces travaux ont fait l'objet d'une publication au workshop WebNLG à INLG2020. A travers le travail sur la verbalisation des triplets RDF, nous avons fait allusion à l'impact de l'évaluation et en particulier à l'impact des métriques d'évaluation automatique pour les tâches de génération de texte. De plus, le problème de la généralisation des modèles supervisés y est mis en évidence.

Dans la partie II de ce manuscrit et la section 3 de ce résumé en français, on détaille les expériences sur une autre tâche de génération de texte : la génération de paraphrases.

La génération de paraphrase peut être vue comme la traduction d'une phrase en une autre phrase de la même langue. C'est pourquoi, historiquement, les modèles de génération de paraphrases sont dérivés des modèles de traduction automatique. Dans un premier temps, nous présenterons la tâche de génération de paraphrases. Ensuite, nous présenterons les expériences sur la génération de paraphrases.

D'abord, nous présenterons des expériences d'entraînement supervisé de modèles encodeur-décodeur pour la tâche de génération de paraphrases. Dans la littérature, des travaux antérieurs avaient déjà été publiés, mais le manque d'uniformité dans les données et le processus d'évaluation, ainsi que parfois le manque d'informations dans les articles correspondants, ont rendu difficile la reproduction de l'état de l'art.

Nous avons alors exploré des stratégies alternatives basées sur la recherche. Il y avait deux raisons d'explorer cette voie. Premièrement, la tâche de génération de paraphrases peut être vue comme une tâche de réécriture de la phrase d'entrée. En d'autres termes, au lieu de la considérer comme une génération à partir de zéro d'une séquence de mots, elle peut être vue comme une modification d'une séquence déjà existante. Deuxièmement, il existe aujourd'hui de puissantes métriques pour évaluer rapidement les paraphrases candidates. En particulier, de grands modèles de langue pré-entraînés fournissent de bons

scores de similarité sémantique et de correction syntaxique que nous avons exploités pour créer une fonction de score de paraphrase. Dans ce cadre, nous avons développé un générateur de paraphrases candidates faiblement supervisé, composé d'un générateur de paraphrases qui utilise des transformations locales applicables à la phrase d'entrée ; et d'un algorithme de recherche qui, en explorant cet espace, sélectionne les paraphrases candidates les plus prometteuses. L'étude des stratégies de génération de paraphrases basées sur la recherche fait l'objet d'une publication dans un atelier à NeurIPS 2020.

Les méthodes basées sur la recherche sont normalement plus générales. Contrairement aux modèles supervisés, elles ne nécessitent pas de corpus de données alignés. En revanche, c'est un mécanisme plus lourd et la génération d'une paraphrase est consommatrice de ressources. D'autre part, nous faisons le pari que les paraphrases générées peuvent être utiles pour améliorer l'apprentissage des modèles supervisés. Nous avons donc étudié la distillation de modèles basés sur la recherche par des modèles supervisés. Toutes les expériences sur la génération de paraphrases ont été publiées dans un article long à EACL 2021.

2 Première Expérience de Génération de Texte : le Challenge WebNLG

La tâche du challenge WebNLG consiste à générer du texte à partir de données RDF en anglais. Les données d'entraînement disponibles pour le challenge sont un ensemble de paires données/texte. Les données sont des ensembles de triplets RDF extraits de *DBpedia* et le texte correspondant est une verbalisation des triplets.

Sur le site officiel du challenge [2], les organisateurs fournissent l'exemple présenté dans la Figure 1. Dans cette figure, les données représentées comme un ensemble de trois triplets RDF sont associées à une phrase.

La tâche RDF-to-text consiste à mettre en correspondance un ensemble de triplets avec l'une de ses lexicalisations possibles. Il est à noter que la verbalisation générée doit être syntaxiquement et sémantiquement correcte. De plus, elle doit contenir le même niveau d'information que les données en entrée.

Pour le challenge, Gardent, Shimorina et al. [2017a] a publié un jeu de données. Une première version des données a été introduite en 2017 avec le challenge [Gardent, Shimorina et al., 2017a]. Une version améliorée des données a été publiée avec la version 2020 du challenge [3].

2.1 Données

Apprentissage

Les données d'apprentissage se composent de 13211 ensembles de triplets uniques associés à plusieurs lexicalisations possibles. Un ensemble de triplets contient jusqu'à sept triplets RDF. Intuitivement, un ensemble contenant plus de triplets sera associé à plus de verbalisations possibles. Nous avons formé

2. https://webnlg-challenge.loria.fr/, dernier accès : 09/02/2021
3. https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/release_v3.0

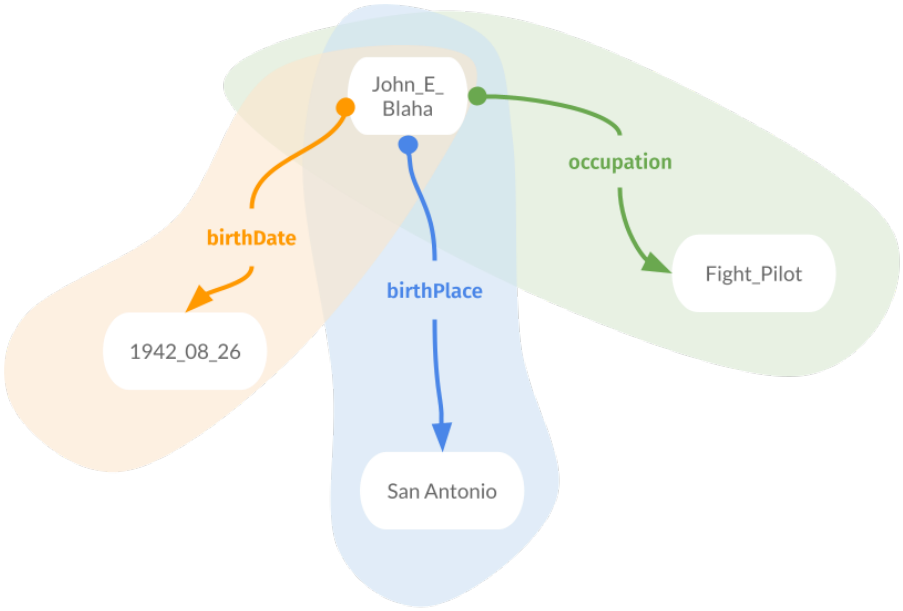


FIGURE 1 – **Exemple de verbalisation de triplet RDF.** Les données sont encodées en triplets RDF. Dans cet exemple particulier, chaque triplet contient le même sujet : *John_E_Blaha*. Les triplets contiennent également les prédicats *birthDate*, *birthPlace* et *occupation* et les objets associés *1942_08_26*, *San_Antonio* et *Fight_Pilot*. L'objectif de la tâche de génération rdf-to-text est de faire correspondre un ensemble de triplets RDF à un texte. Dans cet exemple, l'ensemble de triplets est converti en *John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot.*

des paires RDF/texte associant chaque lexicalisation possible à son ensemble de triplets d'entrée. Cela conduit à un ensemble d'entraînement de 35426 paires. Il est important de souligner qu'une lexicalisation n'est pas nécessairement une phrase unique.

Test

L'ensemble de test fourni pour le challenge WebNLG a été divisé en trois types de données qui peuvent être considérées comme trois tâches différentes pour nos modèles.

Entités et catégories connues. La première partie contient des triplets RDF qui contiennent des entités et des catégories vues dans les données d'apprentissage.

Entités inconnues. La deuxième partie contient des triplets RDF avec de nouvelles entités dans les mêmes catégories que les données d'apprentissage. Par exemple, la catégorie *Astronaute* est présente dans l'ensemble d'apprentissage mais l'entité *Yuri Gagarin* n'est jamais vue.

Catégories inconnues. La troisième partie de l'ensemble de test contient des ensembles de triplets RDF basés sur de nouvelles catégories non présentes dans les données d'apprentissage.

Les modèles soumis sont évalués à l'aide de BLEU [Papineni et al., 2001], METEOR [Banerjee et Lavie, 2005], chrF++ [Popović, 2017], TER [Snover et al., 2006], du BERTscore [Zhang* et al., 2020] et BLEURT. [Sellam et al., 2020].

2.2 Approches Proposées

Il est difficile d'exploiter les architectures de réseaux de neurones profonds pour cette tâche car le nombre de paramètres à entraîner nécessite une grande quantité de données et la verbalisation RDF manque de données alignées.

Les progrès récents des grands modèles pré-entraînés et de l'architecture Transformer sont très prometteurs pour les tâches de génération de texte. Pour le Challenge WebNLG 2020, nous avons proposé d'explorer l'effet du pré-entraînement massif et du modèle Transformer pour la tâche RDF-to-text.

Nous avons choisi d'utiliser des corpus externes pendant le pré-entraînement pour atteindre une meilleure généralisation. Nous construisons deux ensembles de données supplémentaires dans le but de pré-entraîner par réduction de bruit (ST1) et d'augmentation des données (WS1).

En outre, nous avons appliqué le paradigme de *curriculum learning* (CL) pour une convergence plus rapide et de meilleurs minima locaux à travers le processus d'optimisation. Pour rappel, l'idée du CL [Y. Bengio, Louradour et al., 2009] est d'augmenter progressivement la complexité des données pendant l'apprentissage pour imiter le comportement d'apprentissage humain. Dans notre cas, nous assimilons la complexité comme le nombre de triplets RDF en entrée. Plus le nombre de triplets est élevé, plus la complexité est élevée. Nous trions les données WebNLG de telle sorte que les exemples les plus faciles viennent en premier et les exemples les plus difficiles ensuite.

Nous avons entraîné 8 modèles Transformers correspondant à 8 configuration différentes. Chaque modèle est ensuite évalué à l'aide de l'ensemble de tests officiel du challenge WebNLG.

Pour la pré-entrainement par réduction de bruit, les phrases transformées (voir section 2.3.2) ont été directement introduites dans le Transformer. Le modèle doit donc reconstruire les phrases incomplètes. Des exemples d'échantillons prétraités pour les différentes configurations d'apprentissage sont présentés dans le tableau 2.2.

2.3 Résultats

Pour l'évaluation, nous avons utilisé le script d'évaluation officiel de WebNLG (disponible sur github).

Pour évaluer l'impact des différents paramètres que nous voulons explorer, nous avons réalisé une étude d'ablation (*ablation study*). Nous avons défini notre modèle de référence (*baseline*) comme le Transformer entraîné uniquement sur le jeu de données WebNLG, sans curriculum. Nous voulions évaluer l'influence du pré-entraînement, de l'augmentation des données et du CL par rapport à la baseline.

Dans le tableau 2.3, en comparant notre baseline avec les stratégies de pré-entraînement pour chaque catégorie (sans curriculum), nous notons une augmentation moyenne de 3,07, 19,6 et 10,97 respectivement en BLEU lors du pré-entraînement. Des variations similaires peuvent être notées avec les métriques METEOR, chrf++ et BLEURT, bien que moins frappantes. Basée sur les incorporations contextuelles de BERT,

BLEURT donne une bonne estimation de la corrélation sémantique entre la prédiction et les références. La plupart du temps, les métriques basées *n-gram* et sémantiques montrent une parfaite harmonie. Les meilleures valeurs de BLEURT sont obtenues pour les mêmes modèles que les métriques basées *n-gram*. Par conséquent, toutes les métriques tendent à être corrélées, ce qui prouve une bonne concordance. Le gain de performance sur des domaines inconnus s'explique facilement par la diversité des données augmentées. Les nouvelles entités et le lexique spécifique au domaine rencontré aident mieux à modéliser les relations entre les données hors distribution. Ainsi, ces résultats soulignent l'utilité des corpus externes et renforcent le besoin d'un modèle pré-entraîné pour la lexicalisation de triplets RDF.

Sur les catégories connues, notre baseline donne un score de 55,24. Cependant, pour la génération hors domaine, tous les modèles démontrent de graves lacunes. Testé sur des entités inconnues, notre baseline montre une baisse de 42,34 pour atteindre 12,9. Nous constatons une perte similaire et même plus importante dans les catégories inconnues. Dans le cas où les prédicats sont inconnus du modèle, il est difficile de générer une description cohérente des RDFs d'entrée. Des catégories connues aux catégories inconnues, notre baseline est presque pénalisée par un facteur de 5. Un tel effet est tempéré par nos approches de pré-entrainement. La baisse moyenne de la qualité des modèles pré-entraînés (sans apprentissage par programme) est de 25,6 % des catégories vues aux entités non vues, et de 35,5 % des catégories vues aux catégories non vues.

Lorsqu'une approche d'apprentissage avec CL est utilisée, nous assistons à des baisses de performance. Ceci est contre-intuitif et opposé à l'expérience précédente sur notre ensemble de validation.

Il est intéressant de noter que les meilleurs résultats sont révélés avec un pré-entraînement sur ST1, exclusivement. Avec 5 fois moins de données, ST1 conduit à de meilleures performances. Les triplets extraits comprennent sûrement des triplets inexacts. Malgré la qualité imparfaite du jeu de données ST1, son utilisation contribue à la capacité de généralisation.

En revanche, nous constatons que le pré-entrainement par réduction de bruit ne donne pas de résultats satisfaisants lorsqu'il est combiné à notre pré-entrainement sur le jeu de données ST1, ce qui conduit finalement à un effet négatif. Cela peut être dû à une divergence de distribution des entrées entre WS1 et ST1. Le pré-entraînement par réduction de bruit ne nécessite pas de triplets en entrée mais une phrase bruitée. Le décalage entre cette représentation et la linéarisation des triplets peut être la raison de cet effet négatif.

3 Étude de la Génération de Paraphrase

La génération de paraphrases est une tâche fondamentale du traitement automatique des langues. Elle peut être considérée comme une variante de la traduction automatique où la traduction est effectuée dans la même langue que l'entrée. Par conséquent, les progrès de la génération de paraphrases sont étroitement liés à ceux de la traduction automatique. La paraphrase est cependant beaucoup moins étudiée. La raison principale est probablement le manque de ressources disponibles. En particulier, un exemple de modèle performant pour la traduction automatique est le réseau de neurones supervisé, entraîné sur d'énormes bases de données alignées. De tels ensembles de données ne sont pas disponibles pour la génération de paraphrases.

Données

Il n'existe pas d'ensemble de données génériques de paraphrases alignées. Les corpus de paraphrases alignées disponibles sont souvent orientés vers des problèmes spécifiques comme la réponse à des questions ou le sous-titrage d'images. Nous avons utilisé cinq corpus de paraphrases pour nos expériences. Tout d'abord, nous avons utilisé 3 corpus alignés construits pour l'identification des paraphrases : MSRPARAPHRASE, PAWSet QQP. Deuxièmement, deux autres corpus qui ont été construits dans un autre but : MSCOCOpour *le sous-titrage d'images* et OPUSPARCUS pour *la traduction de sous-titres*.

Évaluation Les paramètres d'évaluation automatique sont similaires à ceux utilisés pour le défi WebNLG présenté dans la section 2, à savoir le BLEUet le score BERT.

3.1 Générateurs de Paraphrases Supervisés

Tout comme la traduction automatique ou la génération de RDF en texte, la génération de paraphrases peut être vue comme un problème de séquence à séquence.

Comme la traduction automatique, la génération de paraphrases a bénéficié des réseaux neuronaux profonds et a évolué vers des architectures efficaces de bout en bout qui peuvent à la fois apprendre à aligner et à traduire [Bahdanau, Cho et al., 2016 ; Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez et al., 2017b]. Plusieurs articles, comme [Cao et al., 2017 ; Prakash, Sadid A. Hasan et al., 2016a], présentent la tâche de génération de paraphrases comme un problème supervisé de séquence à séquence. Nos expériences confirment que cette approche est efficace pour des types spécifiques de paraphrases. Elle est également capable de produire des transformations à relativement longue portée et une structure syntaxique complexe, mais elle nécessite d'énormes ensembles de données alignées de phrases de bonne qualité pour l'apprentissage.

Nous avons mené des expériences pour reproduire les résultats des modèles supervisés d'encodeurs-décodeurs tels que rapportés dans la littérature. Il n'existe cependant pas de configuration d'expérience uniforme permettant de comparer directement les modèles et les expériences présentés dans les différents articles. Cela entraîne des problèmes de reproductibilité.

En plus d'essayer de reproduire les résultats existants, nous avons mené les expériences en utilisant un cadre uniforme afin de comparer réellement les résultats et de les étendre à tous les ensembles de données.

Comme baselines supervisées, nous avons entraîné trois architectures de réseaux de neurones réputés comme performants ur MSCOCO et QQP, en particulier, l'architecture Seq2Seq, une architecture *Residual* LSTM [Prakash, Sadid A Hasan et al., 2016b] et un modèle TRANSFORMER [Egonmwan et Chali, 2019a]. Nous avons étendu les expériences aux autres corpus alignés : MSRParaphrase, OPUSPARCUS et PAWS.

3.2 Approches Basées Recherche

En l'absence d'ensembles de données alignées génériques, il reste difficile d'entraîner des modèles de paraphrase génériques de manière supervisée. Pour cette raison, nous avons étudié une approche différente : un schéma de génération basé recherche où des paraphrases candidates sont générées par des

transformations successives à partir de la phrase originale. Les approches basées recherche permettent un meilleur contrôle de la sortie. Elles sont souvent utilisées pour intégrer des contraintes dans la génération de texte comme dans CGMH [Miao et al., 2018].

Nous modélisons la génération de paraphrases comme une séquence d'éditions et de transformations d'une phrase source en sa paraphrase. Nous ne considérons que les transformations locales, c'est-à-dire le remplacement de certains mots ou groupes de mots par d'autres qui ont la même signification ou une signification similaire.

Le principe est simple. A partir d'une phrase source, que nous souhaitons paraphraser, nous créons un générateur de paraphrases candidates en utilisant des transformations locales applicables à la phrase source. Ce générateur peut être vu comme générant un arbre de phrases candidates, où chaque nœud est une paraphrase possible de la phrase source et chaque arête est une transformation effectuée pour aller du nœud source au nœud cible. La racine de l'arbre étant la phrase source, plus on descend dans l'arbre, plus la paraphrase candidate est différente de la phrase source. Dans un deuxième temps, nous recherchons la meilleur candidate ou le meilleur nœud de l'arbre en utilisant un algorithme de recherche.

La base de données *Paraphrase Database* (PPDB) [Pavlick et al., 2015] est une grande collection de règles de paraphrase annotées qui a été construite automatiquement. En appliquant itérativement ces règles à partir d'une phrase source, on obtient un vaste treillis de paraphrases candidates. Certains de ces candidates sont bien formées, mais beaucoup sont syntaxiquement incorrectes.

MCPG

Suivant l'idée de [Chevelu et al., 2009], nous avons d'abord expérimenté *Monte-Carlo Tree Search* (MCTS) pour explorer le trellis. Notre modèle *Monte-Carlo Paraphrase Generator* (MCPG) utilise MCTS pour la tâche de génération de paraphrases. MCTS a besoin d'une fonction de récompense pour mettre à jour sa politique à chaque epoch.

Pour classer les paraphrases candidates, nous avons développé une fonction de score de paraphrase que nous essayons d'optimiser pendant la recherche. L'algorithme MCTS n'est pas conçu pour les problèmes multi-objectifs. Nous avons donc dû combiner trois critères, à savoir la similarité sémantique, la correction syntaxique et la diversité de surface en un seul critère.

L'équilibre entre ces critères est difficile à obtenir. L'option la plus simple était une combinaison linéaire, mais après une analyse quantitative de la distribution des scores, nous avons réalisé qu'il était facile de maximiser le score en appliquant simplement beaucoup d'éditions à la phrase source.

Nous avons donc opté pour le polynôme suivant :

$$\alpha \cdot \text{BERT}_\text{S} + \beta \cdot \text{Lev}_\text{S} \cdot \text{BERT}_\text{S} - \gamma \cdot \text{GPT2} \tag{1}$$

Notre fonction de score permet d'obtenir un assez bon équilibre entre les critères. Cependant, la sortie de MCPG est parfois imparfaite. En fait, MCPG n'est pas vraiment adapté à un problème multi-objectif tel que la génération de paraphrases.

PTS

MCTS est un algorithme efficace sur les problèmes combinatoires où l'évaluation n'est possible que sur les feuilles de l'arbre. Dans notre cas particulier où tous les nœuds sont des feuilles, nous ajoutons une

complexité supplémentaire à la recherche en appliquant de nombreuses transformations au nœud source pour l'évaluer, alors que nous pourrions l'évaluer directement sans faire de roll-out car il est déjà une feuille.

De plus, MCPG ne peut optimiser qu'un seul score. Comme nous voulons évaluer les paraphrases candidates selon trois axes, nous avons dû "scalariser" les différents sous-scores et fixer les poids afin de définir une fonction de score. Ce travail est fastidieux. Les hyperparamètres α, β et γ de (1) sont difficiles à optimiser automatiquement.

Nous avons donc développé un nouvel algorithme de recherche pour résoudre ces deux problèmes : *Pareto-Tree Search* (PTS) qui exploite le concept de *front de Pareto*. Dans notre cas, nous avons un ensemble de paraphrases qui sont notées avec trois sous-scores. Notre lot de candidats est un ensemble fini $X \subseteq \mathbb{R}^3$. Une paraphrase A est *efficace dans le sens de Pareto* s'il n'existe pas d'autre paraphrase B telle que tous les sous-scores de B sont supérieurs aux sous-scores de A. L'ensemble des paraphrases du lot qui sont efficaces constitue ce que l'on appelle le *front de Pareto*. Les autres paraphrases sont dites *dominées*. Nous avons donc adapté MCPG pour explorer le treillis de paraphrases et récupérer une approximation du front de Pareto, en reportant l'équilibre entre les critères comme une étape de post-optimisation rapide. Il s'agit de l'algorithme *Pareto Tree Search* (PTS).

3.3 Résultats

Nous comparons nos modèles avec CGMH un autre algorithme basé recherche présenté par Miao et al. [2018]. Cette ligne de base a une approche basée sur la recherche globalement similaire pour la génération de paraphrases. Nos méthodes MCPG et PTS obtiennent de meilleurs résultats que la baseline CGMH sur tous les corpus, sauf sur MSCOCO où les résultats sont similaires.

En comparaison avec les stratégies supervisées, les résultats sont mitigés. Il est cependant important de garder à l'esprit que contrairement aux stratégies supervisées qui sont réentraînées pour chaque jeu de données, les paramètres des modèles CGMH, MCPG et PTS restent inchangés.

Sur les jeux de données MSCOCO et QQP, les modèles supervisés obtiennent des résultats nettement meilleurs, mais MCPG et PTS obtiennent de meilleurs résultats sur OPUSPARCUS et PAWS à l'exception du BERTscore pour lequel le modèle TRANSFORMER obtient des résultats similaires.

Ces résultats prouvent que même sans ensembles d'entraînement spécialisés, les méthodes génériques basées sur la recherche sont compétitives pour la génération de paraphrases. Cependant, il est un fait que les réseaux encodeurs-décodeurs ont d'excellentes performances pour la génération de textes et ont le potentiel de générer des paraphrases plus complexes que celles obtenues par de simples transformations locales comme dans nos modèles.

L'entraînement d'un réseau de génération de paraphrases général nécessiterait un énorme volume de données. Et il y a encore beaucoup moins de corpus alignés disponibles pour la paraphrase que pour la traduction.

Les méthodes génériques basées sur la recherche peuvent être utilisées comme un modèle hors ligne pour l'augmentation des données.

3.4 Expérience de Distillation

Nous avons mené une expérience simple pour essayer de distiller les connaissances apprises à partir des politiques de recherche MCPG et PTS avec un réseau neuronal supervisé, à savoir un TRANSFORMER.

Une option qui peut être considérée comme un *apprentissage par transfert* est d'enrichir l'ensemble d'apprentissage d'un TRANSFORMER avec les résultats des méthodes basées recherche.

Pour tester cette idée, nous avons utilisé nos modèles MCPG et PTS pour augmenter l'ensemble d'apprentissage d'un TRANSFORMER.

MSRPARAPHRASE contient des paires de paraphrases et des paires de non-paraphrases. Pour nos expériences de génération de paraphrases, nous n'avons eu besoin que des paires de paraphrases, nous avons donc mis de côté les autres paires. Cependant, les paires de non-paraphrases appartiennent à la même distribution de données que les paires de paraphrases utilisées pour entraîner l'encodeur-décodeur supervisé sur MSRPARAPHRASE. Nous utilisons ces paires de non-paraphrases inutilisées pour l'entraînement de notre expérience de distillation.

Tout d'abord, nous transformons les paires de non-paraphrase de MSRPARAPHRASE en phrases sources à paraphraser par les modèles que nous voulons distiller, à savoir MCPG et PTS.

Ensuite, nous échangeons la paraphrase générée avec la phrase source pour créer de nouvelles paires de paraphrases. Les paraphrases générées deviennent les nouvelles phrases sources. Et les phrases sources initiales deviennent les nouvelles phrases de référence. Cette technique, appelée *back-translation*, a été introduite par EDUNOV et al. [2018] et SENNRICH et al. [2016]. Elle garantit que le modèle a toujours une phrase syntaxiquement correcte comme référence de sortie. En effet, comme un encodeur-décodeur est entraîné par *teacher forcing* sur la phrase de référence, l'apprentissage avec une référence bruitée peut dégrader l'apprentissage.

Avec ces paires de paraphrases nouvellement générées, nous avons augmenté l'ensemble d'apprentissage MSRPARAPHRASE . Nous avons ensuite formé de nouveaux modèles supervisés TRANSFORMER sur les ensembles de train augmentés, l'un augmenté de MCPG et l'autre augmenté de PTS.

Les résultats montrent que l'augmentation de l'ensemble d'apprentisage de MSRPARAPHRASE par des paires de paraphrases générées par MCPG et PTS a augmenté le score BLEU de 5 points. En revanche, le BERT score a légèrement diminué.

Malgré ces résultats surprenants concernant le BERT score, les approches basées recherche semblent être très intéressantes en tant que système d'augmentation des données hors ligne pour surmonter les problèmes des encodeurs-décodeurs. Elle offre un compromis entre les approches supervisées et faiblement supervisées.

4 Conclusion et Perspectives

Le modèle encodeur-décodeur est au cœur du travail présenté dans cette thèse. Cette famille de réseaux neuronaux construite pour des tâches de transformation de séquences est bien adaptée aux tâches de génération de texte comme la lexicalisation de données ou de paraphrases ; et a amélioré l'état de l'art dans de nombreuses tâches de génération de texte. Les performances du modèle Transformer ont conduit au développement de modèles pré-entraînés, tels que les *embeddings* contextuels BERT et le modèle de

langue GPT-2. Le Transformer est également largement utilisé comme modèle supervisé appris à partir de zéro pour des tâches de génération de texte sur des corpus alignés et surpasse souvent le Seq2Seq original qui repose sur des RNN.

En suivant cette idée, nous avons exploré la tâche de verbalisation de données RDF. Nous avons entraîné des modèles supervisés de Transformer sur une version récemment publiée du jeu de données WebNLG et avons étudié en profondeur plusieurs stratégies de pré-entraînement pour surmonter la petite taille du corpus aligné.

Le domaine de la génération de paraphrases est étroitement lié à la traduction automatique en raison de la nature de la tâche. Cependant, cette dernière a reçu beaucoup plus d'attention en raison de la disponibilité de corpus alignés massifs. Pour la génération de paraphrases, les corpus sont plus petits et plus biaisés.

Dans cette thèse, nous avons entraîné les modèles Transformer sur des corpus alignés pour les comparer directement avec le modèle de littérature qui était basé sur des RNNs à l'époque. Ceci a conduit à deux problèmes. Premièrement, nous avons trouvé qu'il était très difficile de reproduire les résultats de l'état de l'art pour la tâche de génération de paraphrases. Une contribution importante de la thèse a été de proposer un cadre expérimental uniforme pour comparer les modèles encodeur-décodeur pour la génération de paraphrases et d'étendre l'entraînement à 5 corpus de paraphrases alignés, y compris les deux corpus très biaisés -mscoco et quora- souvent utilisés dans la littérature. Deuxièmement, les modèles Transformer n'ont pas donné les résultats escomptés.

En effet, nous n'avons noté aucune différence significative entre les architectures de réseaux neuronaux basées sur les RNN et les Transformers, sauf pour MSRParaphrase. Nous supposons qu'étant donné que MSRParaphrase est plus petit et contient des phrases plus longues, alors le Transformer s'adapte moins bien que les réseaux neuronaux basés sur les RNN.

En raison de la nature de la tâche et des données disponibles pour la formation supervisée, nous avons exploré la voie des stratégies basées recherche pour générer des paraphrases. La motivation principale était de fournir un meilleur contrôle de la paraphrase générée.

Les résultats ont montré que sans un ensemble d'entraînement spécialisé, les méthodes génériques basées sur la recherche sont compétitives pour la génération de paraphrases, mais il est clair que les modèles supervisés ont le potentiel de générer des paraphrases plus complexes.

Enfin, comme les méthodes basées sur la recherche ne pouvaient être utilisées que comme modèle *offline* pour l'augmentation de données, nous avons mené des expériences de distillation. Ces expériences ont montré que la distillation améliorait les performances du modèle TRANSFORMER entraîné sur MSR-Paraphrase et constitue une branche de recherche prometteuse.

Afin d'approfondir cette idée de pré-entraînement et de distillation. Une perspective de ce travail pourrait être de *fine-tuner* un modèle de langage générique pré-entraîné pour la tâche de génération de paraphrases.

Une autre perspective de ce travail serait de développer de nouvelles politiques de recherche dans le cadre de l'approche basée recherche. En particulier, on pourrait penser à une politique entièrement apprise grace à l'apprentissage par renforcement. La principale difficulté consiste à trouver un bon oracle pour entraîner la politique. Nous supposons que cela sera difficile, mais c'est une piste de recherche intéressante.

En outre, on pourrait ajouter plus de contrôle au schéma supervisé en explorant les solutions de prédiction structurée pour la génération de texte.

Introduction

Introduction

In the last decade, deep learning and neural networks have offered new tools that have improved the state-of-the-art in many text generation tasks such as machine translation. The recent performance gains in text generation have even been the subject of general media articles calling it a *"revolutionary artificial intelligence system"*[4].

Similar to what happened in the field of computer vision with convolution layers, the field of Natural Language Processing (NLP) has seen the beginning of a new era with the development of neural networks. In particular, there are three important milestones.

Firstly, continuous embedding models such as *word2vec* [Tomas Mikolov, 2012] and *GloVe* [Pennington et al., 2014] replace the then widely used one-hot encodings and bag-of-words as distributed vector representations of words. These embeddings are generated by neural networks that learn a space where the distance between words is related to semantic similarity.

Then, the encoder-decoder models from the *Seq2Seq* [Cho et al., 2014; Sutskever et al., 2014] to the *Transformer* [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al., 2017a] are end-to-end architectures for generating text from an input sequence. For example, for machine translation, an encoder-decoder model is fed with a sentence in one language and generates its translation into another language. These models achieved significant improvements in many text generation tasks.

Finally, very recently – at the same time as the work presented in this thesis –, the field of NLP has witnessed a major paradigm shift with the release of huge pre-trained language models Devlin et al. [2018], Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Veselin Stoyanov, et al. [2020], and Radford et al. [2019] that can be fine-tuned on many text generation tasks.

In this thesis, we focused on the topic of **Search-Based and Supervised Text Generation** in the midst of this NLP revolution. Supervised models contain more and more parameters and therefore require more and more aligned data to be trained. The necessary data is not yet available for several tasks such as RDF triples verbalization or paraphrase generation. We explored these two tasks. Also, neural networks are efficient but not perfect. The race for performance raises questions about evaluation and in particular the automatic evaluation metrics used to evaluate supervised models. We have encountered obstacles in the reproducibility of some models – and thus results – published in the literature.

We therefore studied supervised models for text generation and also explored alternative search-based models.

This manuscript is divided into two parts. The first part can be seen as an introductive part.

As the encoder-decoder model is at the heart of our work, it is an essential prerequisite for reading the work presented in this manuscript. Although there are many good articles, books or blog posts that

4. "New AI fake text generator may be too dangerous to release, say creators", Alex Hern, The Guardian, 14/02/2019, https://www.theguardian.com/technology/2019/feb/14/elon-musk-backed-ai-writes-convincing-news-fiction

introduce the *Seq2Seq* model and its derivates, we give a chronological presentation of the development of these models in Chapter 1. Chapter 1 does not pretend to be an exhaustive course on the subject and we refer the reader to pointers when necessary.

A good way to explore the latest advances in supervised models is to apply them to a particular task. This is what we did by participating in the WebNLG Challenge. The WebNLG Challenge [Gardent, Shimorina, et al., 2017b] aims at promoting the verbalization of RDFs triples. Indeed, the previous edition of the challenge having taken place in 2017, the models previously submitted for the challenge did not rely on Transformers and even less on large-scale pre-training strategies. For the 2020 WebNLG Challenge, we submitted Transformers trained using data augmentation and pre-training strategies. We report the experiments in Chapter 2. This first experiment with text generation gives us the opportunity to introduce the data preprocessing and learning pipelines. This is why we have chosen to leave the Chapter 2 in the first introductory part of this thesis. However, we highlight that the work presented in Chapter 2 is novel and has been the subject of a publication to the WebNLG Workshop at INLG2020. It should also be mentioned that the work on the WebNLG challenge was carried out in collaboration and equal contribution with another PhD student, Sébastien Montella. Through the work on the verbalization of RDF triples, we hint at the impact of evaluation and in particular of automatic evaluation metrics for text generation tasks. Also, the generalization problem of supervised models is highlighted.

Part II of this manuscript reports experiments on another text generation task : paraphrase generation. This part is divided into four chapters.

Generating a paraphrase of a sentence can be seen as translating the sentence into another sentence in the same language. That is why, historicaly the paraphrase generation models derived from the machine translation models. In Chapter 3, we introduce the paraphrase generation task and give an overview of the paraphrase generators. The following three chapters reports experiments on paraphrase generation.

In Chapter 4, we present supervised encoder-decoder model training experiments for the paraphrase generation task. In the literature, previous work had already been published, but the lack of uniformity in the data and evaluation process as well as sometimes the lack of information in the corresponding papers made it difficult to reproduce the state of the art.

We then explored alternative search-based strategies. There were two reasons for exploring this path. Firstly, the paraphrase generation task can be seen as a rewriting task of the input sentence. That is, instead of seeing it as a generation from scratch of a token sequence, it can be seen as a modification of an already existing sequence. Secondly, there are now powerful metrics for quickly evaluating candidate paraphrases. In particular, large pre-trained language models provide good semantic similarity and syntactic correctness scores that we leveraged to create a paraphrase score function. In this framework, we have developed a weakly-supervised candidate paraphrase generator consisting of a paraphrase space generator that uses local transformations applicable to the input sentence; and a search algorithm that, by exploring this space, selects the most promising candidate paraphrases. In Chapter 5, we provide details of these search-based methods. The study of the search-based stategies for paraphrase generation is the subject of a publication in a workshop at NeurIPS 2020.

Search-based methods are normally more general as they are not supervised on aligned data corpora

like supervised models. On the contrary, it is a more cumbersome mechanism and the generation of a paraphrase is resource consuming. On the other hand, we hypothesize that the generated paraphrases can be useful to improve the learning of supervised models. We have therefore studied the distillation of search-based models by supervised models. The results are presented in Chapter 6.

All the experiments on paraphrase generation were published in a long paper at EACL 2021.

Part I

Background and First Experiments

Chapter 1

An Introduction to Encoder-decoder Neural Networks

As outlined in the Introduction Chapter, this thesis is about sequence generation and in particular on the vast topic of text generation. In Chapter 2, we present the work done for the WebNLG Challenge that involves a data-to-text generation task. In Part II, we explore paraphrase generation. Paraphrase generation is very similar to machine translation as it can be seen as a translation into the same language. In Chapter 3, we detail the major differences between the two tasks.

In this Chapter, we introduce the neural network architectures we used to tackle the sequence predictions tasks. In a first Section 1.1, we describe the recurrent neural networks that are the building blocks of the original *Seq2Seq* model presented in Section 1.2. In the following Section 1.3, we focus on the attention mechanism that allowed significant improvements to the *Seq2Seq* for text generation and lead to the Transformer model. The Transformer model and its derivates are presented in Section 1.4. In Sections 1.5 and 1.6, we focus on the learning and inference of such architectures and its limitations.

1.1 The Recurrent Neural Network (RNN)

Recurrent neural networks (RNN) [Rumelhart et al., 1986] are a type of neural networks designed to process sequences $\{x^{(1)}, \dots, x^{(n)}\}$.

An RNN relies on the principle of parameters sharing to deal with variable-length sequences. Indeed, a fully connected layer has fixed size input and output. At the opposite, an RNN will generate a sequence by producing each token of the output iteratively using an update rule applied to the previous outputs.

In Figure 1.1, we display an unfolded RNN cell that maps a sequence x to a generated sequence o that is compared with the target sequence y.

The RNN's output can be seen as a probability distribution over the possible output tokens. For instance, for text generation, the RNN can be used to generate a text word by word. In this simple example, one have a vocabulary of words and the RNN models a probability distribution over the possible words. The generated distribution is then directly compared to the target distribution corresponding to the target word in the ground-truth sentence.

One of the motivations behind RNNs is the possibility to handle long-term dependencies. In fact, at

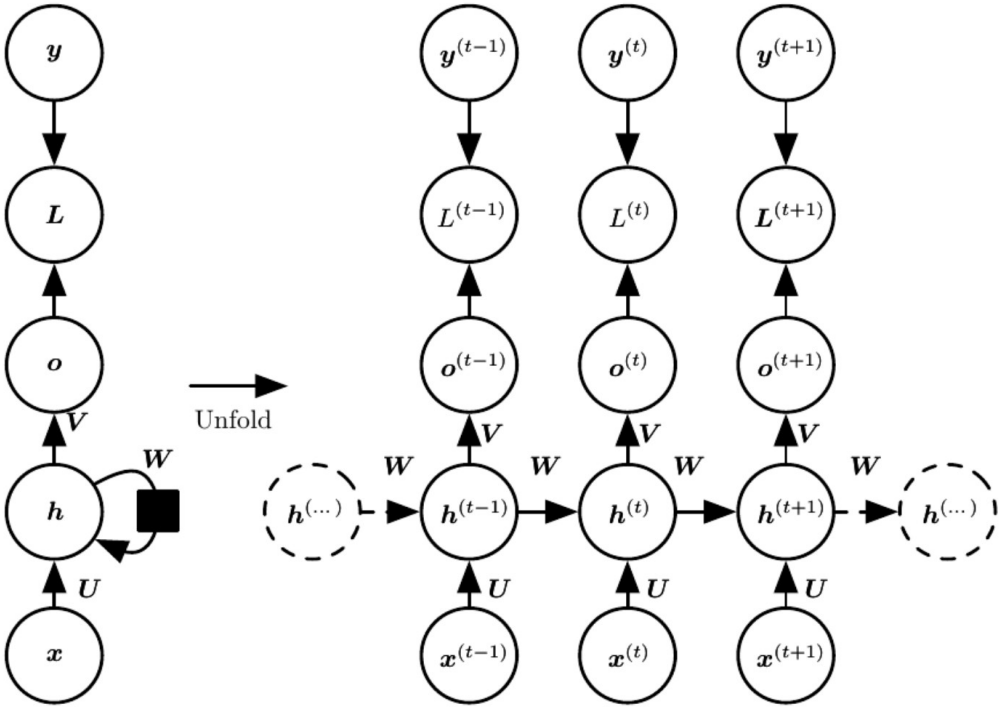


Figure 1.1 – Figure reproduced from Figure 10.3 in [I. Goodfellow et al., 2016]. Computational graph of a recurrent network. The reccurent cell maps iteratively the input x to the output o. The output is compared with the target sequence y to compute the training loss L.

each time step, the generation is connected to all the previous generations. In theory RNNs are able to handle long-term dependencies. But Y. Bengio, Simard, et al. [1994] found that in practice it might be difficult due to the vanishing gradient problem.

Hochreiter and Schmidhuber [1997] introduced a reccurent cell to tackle this issue: Long Short Term Memory networks (LSTM). LSTM are able to remember information for a long time. For more details on the LSTM, the reader is referred to [Hochreiter and Schmidhuber, 1997] or blog post [Olah, 2015].

1.2 The Encoder-Decoder Architecture

RNNs can handle generation of variable size sequences. However the number of output tokens has to be the same as the number of input tokens. In other words, the size of the generated sequence is always the same as the input sequence. While this is no problem for tasks like *Optical Character Recognition* (OCR) or *Voice Activity Detection* (VAD), it is not convenient for many text generation tasks where the input and the output do not have the same size. For instance, in machine translation, a sentence in one language has not always the same size as a its translation to another language.

To tackle this problem, Sutskever et al. [2014] introduced a model called *Sequence-to-Sequence (Seq2Seq)* that combines two RNNs to deal respectively with the variable-size inputs and the variable-size outputs. At the same time, Cho et al. [2014] introduced a similar model and called it *Encoder-Decoder*. In this thesis, we use *Seq2Seq* and *Encoder-Decoder* interchangeably. We display the *Encoder-Decoder* architecture in Figure 1.2

Initially, the *Seq2Seq* model has been developped to tackle machine translation. The goal of the translation task is to find a target sentence y that maximizes the conditional probablity of y given a source sentence x : $argmax_y\ p(y|x)$.

The *Seq2Seq* belongs to the neural machine translation (NMT) framework that uses neural networks to tackle the translation task. In the Encoder-Decoder framework, there are two parts: an encoder that encodes a variable-length source sentence into a fixed-length vector and a decoder that decodes the vector into a variable-length target sentence.

The encoder takes as inputs the source sentence tokens $x = (x_1, \dots x_{T_x})$ and outputs a vector c. The encoder is an RNN such that :

$$h_t^{enc} = f(x_t, h_{t-1}^{enc})$$

and

$$c = q(\{h_1^{enc}, \dots, h_{T_x}^{enc}\})$$

, where $h_t^{enc} \in \mathbf{R}^n$ is the RNN hidden state at time t, and c is a vector generated from the sequence of the hidden states. Sutskever et al. [2014] set c as the last produced hidden state that has been computed according to all previous hidden states.

The decoder is trained to predict each token y_t at a time given c and the previously predicted words $\{y_1, \dots, y_{t-1}\}$. The decoder is also an RNN that models a probability distribution over the generated

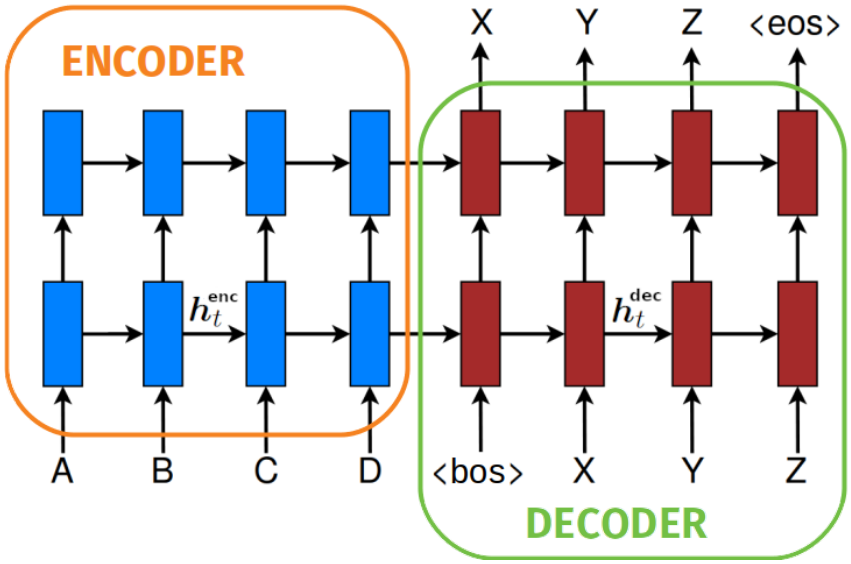


Figure 1.2 – The initial figure is from [Luong et al., 2015]. We added the encoder and decoder frames for visual clarity. The figure illustrates the encoder-decoder architecture. On the left part, there is the two-stacked RNN encoder that takes as inputs the sequence "A B C D". On the right part, there is the two-stacked RNN decoder that generates the sequence "X Y Z". The token <*bos*>("beginning of sentence") indicates to the decoding RNN that the current prefix is empty and the token to generate is the first of the sequence. When the token <*eos*> ("end of sentence") is generated, it means the end of the decoding.

tokens

$$p(y_t|y_1,\dots,y_{t-1}) = g(y_{t-1}, h_t^{dec}, c)$$

The probability over the translation y is obtained by decomposing the joint probability according to the chain rule:

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1,\dots,y_{t-1}\}, c)$$

where $y = (y_1,\dots,y_{T_y})$ is the target distribution.

1.3 The Attention Mechanism

As mentionned in Section 1.1, Y. Bengio, Simard, et al. [1994] highlighted the long-term memory issues of the RNNs. While LSTM was build to tackle the issue, J. Zhao et al. [2020] pointed out that it is difficult to judge by the performances of the LSTM and finally concluded that LSTM do not have long memory.

In practice, for machine translation for instance, one can avoid memorising all the information in the input sentence if one knows where to look at in the latter, at each step of the translation. Bahdanau,

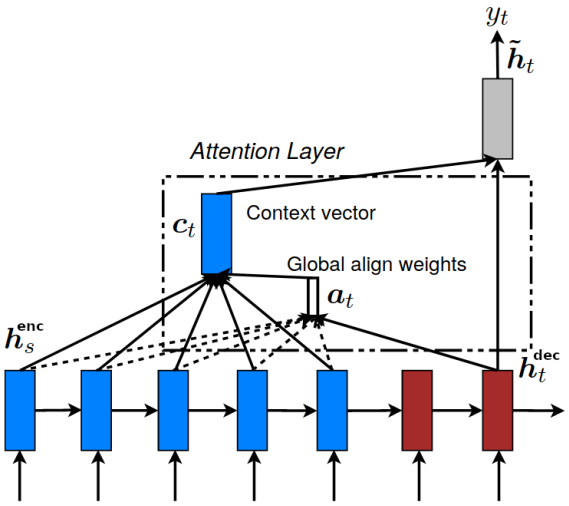


Figure 1.3 – Attention Layer used at each decoding steps. The figure is from [Luong et al., 2015]. The figure illustrates the decoding step t. The computed hidden state h_t is used to compute alignmenent scores that are then used to weight the sum of the encoding hidden states into a context vector c_t. The computed weights allow the decoding to focus on some parts of the input sentence at this time step t.

Cho, et al. [2016]'s idea was to add an alignment step with the input sequence at each decoding step.

Indeed, Bahdanau, Cho, et al. [2016] introduced an alignement system to enhance the *Seq2Seq* architecture that improved its performances on many tasks: the attention mechanism.

In figure 1.3, we illustrate the attention layer added for the decoding step.

The attention layer is the computation of a new context vector c_i that depends on the sequence of annotations to which the encoder map the input sequence, in practice the hidden states $(h_1, \ldots, h_{T_x})$. Each hidden state contains information about the whole sequence with a strong focus on the local parts surrounding the position i.

The context vector c_i is a weighted sum of the annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

The weight α_{ij} associated with each position j is computed through a softmax:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

where the e_{ij} are scores computed by an alignment model that tells how well the inputs around position j and the output at position i match together.

Intuitively, this implements a mechanism of attention in the decoder. The decoder can focus on some parts of the source sentence to generate its current token.

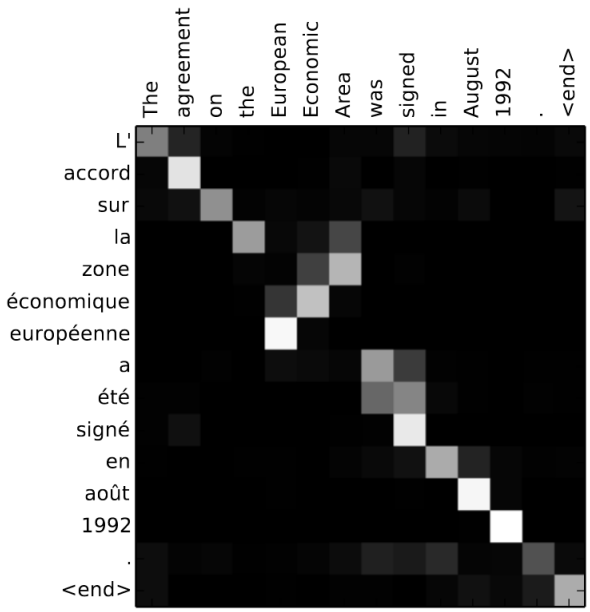


Figure 1.4 – Example of attention matrix for the generation of a French sentence from an English source sentence. The task is machine translation. The figure is from [Bahdanau, Cho, et al., 2016]. This matrix displays the weight α_{ij} of the alignement of the j-th source word in English on the x-axis for the i-th generated word in French on the y-axis. White squares means the weight is high. The diagonal shows that there is almost a litteral word-to-word translation from the source to the target sentence. However, we observe that "the European Economic Area" is translated into "la zone économique européenne", so there is a syntaxe difference between the place of the adjectives in English and French that is highlighted by the computed alignement. Also, we can notice that for the generation of the auxiliar in French "a été" the attention was divided to the two words "was" and "signed" in the source sentence.

In figure 1.4, we display an example of attention matrix drawn from [Bahdanau, Cho, et al., 2016].

This attention mechansim allows the encoder to control which information from the source sentence should be encoded into the fixed-length vector. Indeed the information can be stored throught the sequence hidden states which will be used by the decoder at the right time. As seen in the example displayed in Figure **??**, most of the decoding steps have a strong focus on one word in the source sentence.

While the attention model introduced by Bahdanau, Cho, et al. [2016] is a global attention mechanism, Luong et al. [2015] introduced a local attention mechanism.

The concept of attention is extended to become at the core of the encoder-decoder model in the Transformer model introduced by Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al. [2017a].

1.4 The Transformer and its Derivates

1.4.1 The Transformer

The Transformer introduced by Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al. [2017a] is an encoder-decoder model that replaces the RNN layers using multi-head attention layers. The Transformer model maintains the encoder-decoder structure of the *Seq2Seq* models. It contains an encoder that computes a representation of the input sequence, and a decoder that takes this representation along with the output tokens to autoregressively generate the output sequence. However, in the Transformer the encoder and decoder are composed of stacked self-attention and point-wise, fully connected layers.

In Figure 1.5, we display the architecture of the Transformer model as introduced by Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al. [2017a].

1.4.2 Transformer-based Models

BERT BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2018] is a pre-trained language model based on the Transformer model. In particular, BERT's model architecture is a multi-layer bidirectional Transformer encoder.The BERT model is pre-trained to generate contextual embeddings of the tokens of a sentence.

GPT2 Similarly, GPT-2 is a pre-trained large-scale unsupervised Transformer language model that achieved state-of-the-art performances on several language modeling benchmarks [Radford et al., 2019].

The field of NLP has been significantly impacted by these large pre-trained language models and novel learning architectures which improved dramatically state-of-the-art results on several benchmarks [Clark et al., 2020; Devlin et al., 2018; Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Veselin Stoyanov, et al., 2020; Radford et al., 2019; Z. Yang et al., 2019; Z. Zhang et al., 2019]. The Transformer model [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez, et al., 2017b] is in most cases at the core of these improvements. The self-attention mechanism of Transformer enables a better understanding of underlying dependencies in sentences.

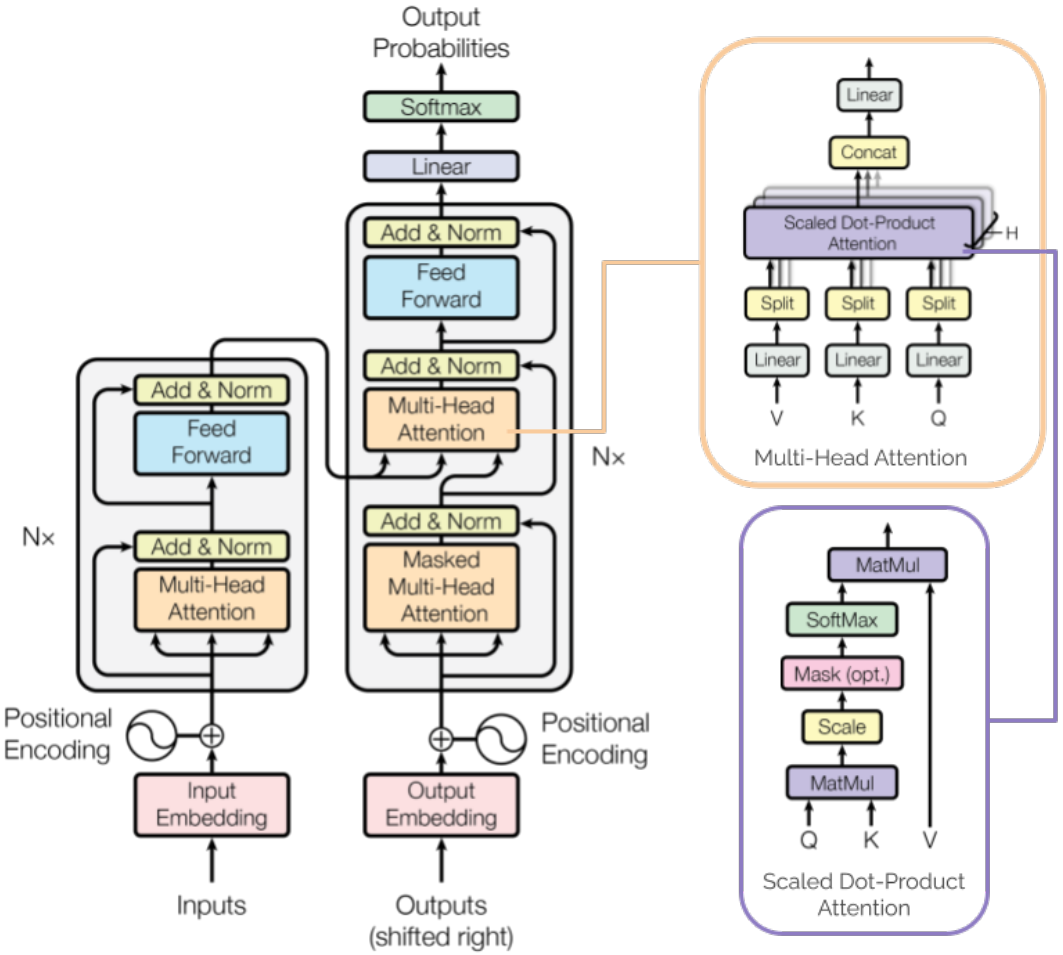


Figure 1.5 – The figure combines 3 figures from [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al., 2017a] to illustrates the Transformer architecture. The main figure on the left illustrates the building components of the encoder and decoder in the Transformer. On the right, we display the multi-head attention cells and the scaled dot-product attention layer.

1.5 Inference

As detailed in Section 1.2, the encoder-decoder model optimizes the probability of the whole target sequence y by decomposing the joint probability with the chain rule. Decoding consists of generating the candidate sequence $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_T)$ that maximizes the joint probability $p(y|x) = \prod_{t=1}^{T} p(y_t|x, \{y_1, \ldots, y_{t-1}\}$, x being the intput sequence. In order to generate the real maximum, we need to generate every path possible in the output space and compute the corresponding probability. This is intractable. As a consequence, we use approximation strategies to decode the candidate sequence. In the following subsections 1.5.1 and 1.5.2, we present two decoding strategies used for inference : *greedy* decoding and *beam search*.

For simplicity, we consider the problem of generating a sequence of letters of size 3. The size of the vocabulary is 4 and the possible letters are : *a,b,c,d*. The space of possible candidates is displayed in Figures 1.6 and 1.7.

It is important to note that this example considers the generation of a sequence of fixed-size. In practice, the decoding strategies are used for variable-length sequences which add complexity. Indeed, for the fixed-size case, we need to consider every path of this size in the output space and chose the best; for the variable-length case, we need to consider every path for every size possible and chose the best.

Also, in text generation tasks, the size of the vocabulary can be several thousand tokens.

1.5.1 Greedy Decoding

The principle of the *greedy* decoding is displayed on Figure 1.6. The idea is simple, it consists of the following approximation : $arg_y \, max \prod_{t=1}^{4} p(y_t|x, \{y_1, \ldots, y_{t-1}\} \approx \prod_{t=1}^{4} arg_{y_t} \, maxp(y_t|x, \{y_1, \ldots, y_{t-1}\}$. This approximation is easy done because it follows how the decoder of the *Seq2Seq* constructs sequences autoregressively.

On Figure 1.6, we can notice that at each decoding step, a lot of potential sequences are discarded. That is why, often we want to consider more that one path at each step for the following steps. It is the idea of the beam search.

1.5.2 Beam Search

The principle of the *beam search* decoding is displayed on Figure 1.7. In the Figure, we only displayed the exploration of the output space during decoding. The parallel with the *Seq2Seq* is the same as in Figure 1.6. The idea of Beam Search is to keep a beam - a set - of the most promising path at each step. In the figure, the beam is of size 2 meaning that we keep the top 2 sequences at each step to keep going the decoding.

1.6 Maximum Likelihood Training (MLE) and its Limitations

The standard training loss for the *Seq2Seq* model is an approximation of the maximum likelihood estimation (MLE). At each step in the sequence, the decoder outputs logits that model a probability

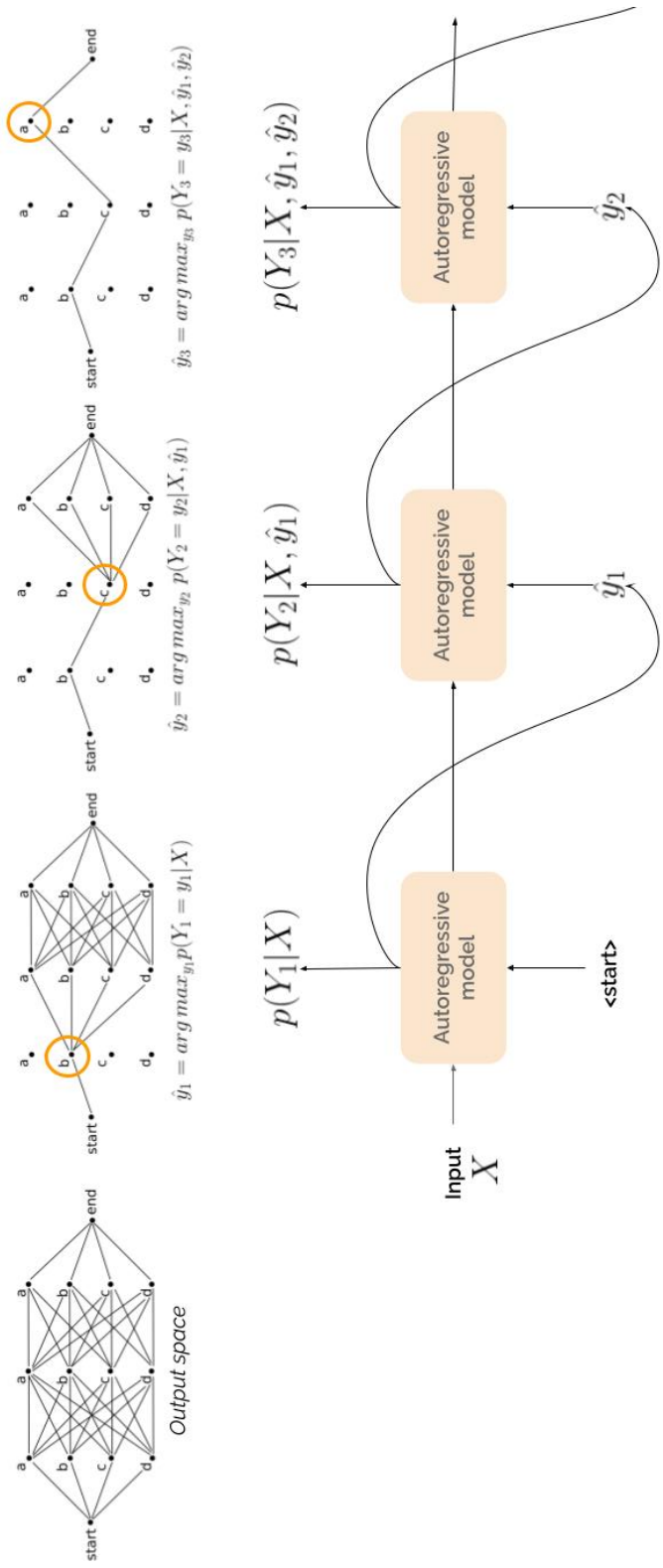


Figure 1.6 – **Inference using a greedy decoding**

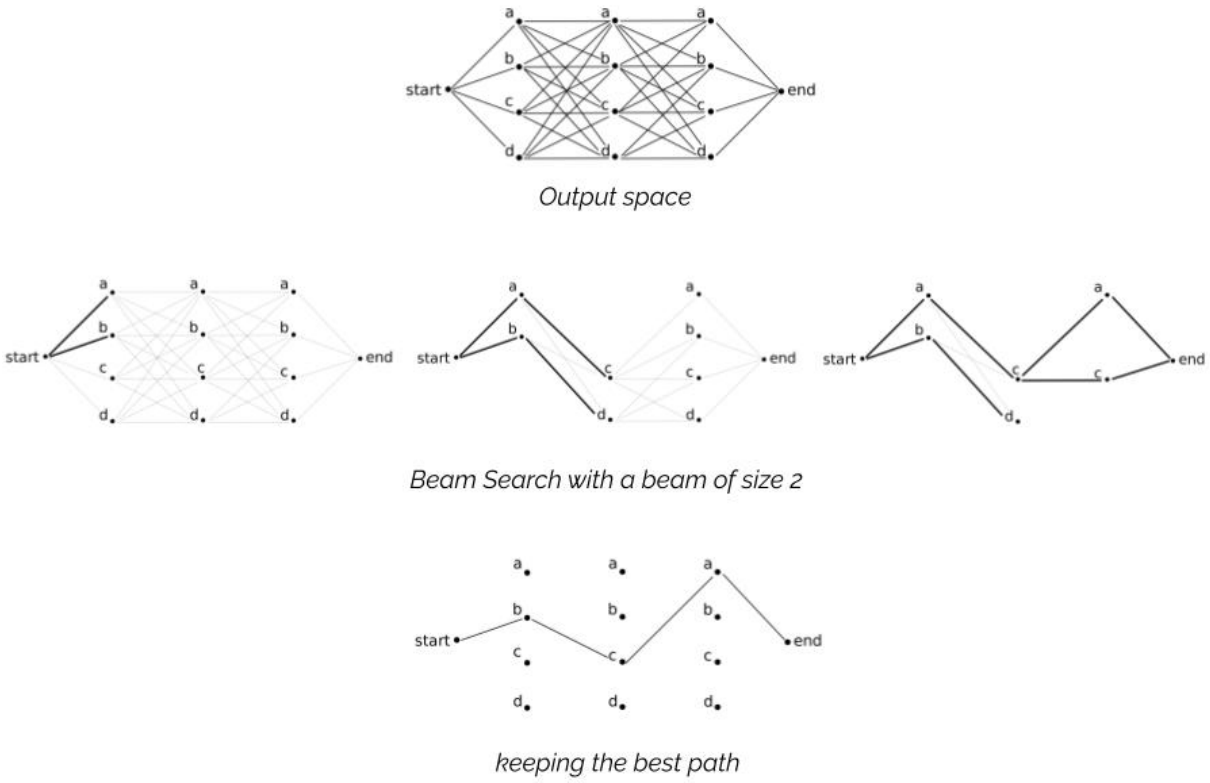


Figure 1.7 – **Inference using a beam search**

distribution over the tokens. We maximize then the probability of the ground truth tokens contained in the target sequence.

The *Seq2Seq* decoder is adapted for structured prediction where we want to model the joint probablity of a target sequence $(y_1, \ldots, y_{T_x})$ given an input x; T_x being the length of the output sequence associated with input x.

To model this, the hidden state generated by the RNN or Transformer decoder is fed through a projection layer (a linear classifier layer) to obtain a vector of scores s_t over the tokens. This scores vector is then normalized using a softmax layer to obtain a distribution o_t over the tokens of the vocabulary. The vector value o_t can be interpreted as the conditional distribution of the t^{th} token given the input and the prefix already generated : $o_t(a) := p(a|y_1, \ldots, y_{t-1}, x)$.

By applying the chain rule, one can obtain the joint probability of the whole sequence y:

$$p(y|x) = p(y_1, \ldots, y_T|x) = p(y_1|x)p(y_2|y_1, x) \ldots p(y_T|y_1, \ldots, y_{T-1}, x) = \prod_{t=1}^{T} o_t(y_t)$$

We emphasize here that at the training stage, the only information available for the model is contained in the ground truth target sequence. The only value that can be optimized is the distribution of the target token given the prefix of the ground truth. In other words, we force the prefix of the generated sequence

to be the same as the ground truth sequence, in order to optimize the likelihood of the target token. This strategy is called **teacher forcing**. While maximum likelihood training is successful in many sequence generation tasks; teacher forcing and MLE has several limitations.

1.6.1 Exposure Bias

When training with teacher forcing, the model learns the probabilities of the tokens with respect to the prefix of the ground truth. But at test time, the model generates the tokens autoregressively with respect to its own previous predictions. This discrepency between the training and inference schemes is called *exposure bias* [Ranzato et al., 2015]. Exposure bias is said to be a major issue with MLE and it has led to the exploration of alternative training strategies [S. Bengio et al., 2015; Chang et al., 2015; Leblond et al., 2018; Ross et al., 2011; Sabour et al., 2018; Schmidt, 2019; Welleck et al., 2019]. In fact this phenomenon is typical for sequential prediction tasks explored in the structured prediction literature Daumé et al. [2009].

1.6.2 Training Loss and Evaluation Metrics Discrepency

Each sequence generation task is associated with corresponding metrics for evaluation. For instance, a standard evaluation metric used for machine translation and many text generation tasks is BLEU (See Section 2.1.4 for a definition). The test errors are very different from the training loss that only focuses on optimizing the probability of the ground truth.

In particular, several papers highlighted that the loss used to train RNNs is local, at the token level [Bahdanau, Brakel, et al., 2016; Ranzato et al., 2015; Sabour et al., 2018; B. Zhang, 2016]. At the opposite, the metrics used to evaluate the trained models are at the sequence level.

Chapter 2

A First Experiment with Structured Prediction : the WebNLG Challenge

The WebNLG challenge aims at promoting the task of automatic RDF-to-text generation.

The RDF (Resource Description Framework) format is an effective format to store information in large-scale Knowledge Bases (KB). However, it is not easily interpretable by humans. As a direct result, the automatic verbalization of RDF triples had gained in popularity. The WebNLG challenge is therefore dedicated to promoting this RDF-to-text generation task.

2.1 A Data to Text Generation Task

In this section we will introduce the data format and the text generation task.

Gardent, Shimorina, et al. [2017a] made available a dataset of data/text pairs in the context of the WebNLG shared task and provided a challenging benchmark for microplanning. Microplanning consists in mapping a given content to a text verbalizing this content Gardent, Shimorina, et al. [2017b]. Initially, this dataset was created with two goals: first, to encourage the development of RDF verbalizers; second for the development of microplanners capable of processing a wide range of linguistic constructions.

2.1.1 Knowledge Bases

A knowledge graph is a type of knowledge base.

Knowledge Base

As defined by Jarke et al. [1989]:
A representation of heuristic and factual information, often in the form of facts, assertions and deduction rules. For Hogan et al. [2021], *knowledge* may be accumulated from diverse sources. It can be composed of statements such as *"Paris is the capital of France"* or *"all capital are cities."*

Knowledge Graph

As defined by Hogan et al. [2021]:
A graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities.

Additional knowledge can be extracted from a knowledge graph using inductive methods. For instance if the knowledge base contains the two previous examples *"Paris is the capital of France"* and *"all capital are cities."*, one can extract the additional knowledge *"Paris is a city"*.

There exist *open knowledge graphs* that are available online. For instance DBpedia [Lehmann et al., 2015] and Wikidata [Vrandečić and Krötzsch, 2014] are extracted from Wikipedia [1].

Directed edge-labelled graphs

As defined by Hogan et al. [2021]:
A set of nodes (like "France", "Paris", "1996-03-03 03:40") and a set of directed labelled edges between those nodes (like "$Paris$" $-$ $city \rightarrow$ "$France$"). Nodes represents entities and edges represents binary relations between the entities.

The *Resource Description Framework (RDF)* [Consortium et al., 2014] is a standard model based on directed edge-labelled graphs for describing Web resources. Knowledge graphs can be represented in RDF.

An RDF statement, or RDF triple, states knowledge by triplets ⟨subject; predicate; object⟩. For instance if we take the triple ⟨*Alan_Bean; birthDate; 1932*⟩, the subject is *Alan_Bean*, the predicate/relation/property is *birthDate* and the object is the date *1932*.

The RDF formalism is used to encode many large datasets :

— *DBpedia* extracts facts from Wikipedia articles and publishes them a RDF data;
— *MusicBrainz* publishes information about Music Albums;
— *FOAF (Friend of a Friend)* is designed to describe people, their respective interests and interconnections;
— *LinkedGeoData*.

The RDF/XML format is an XML-based standard syntax for serializing RDF.

This raw triplet representation is an efficient and simple way to store data in knowledge base, but it is difficult for humans to interpret.

2.1.2 A Text Generation Task

The training data available for the WebNLG challenge [Gardent, Shimorina, et al., 2017a] is a set of data/text pairs. The data are sets of RDF triples extracted from *DBpedia* and the corresponding text is a verbalisation of the triples.

1. www.wikipedia.com

On the official website of the challenge [2], they provide the example shown in Figure 2.1. In this Figure 2.1, the data encoded as a set of three RDF triples is mapped to a sentence.

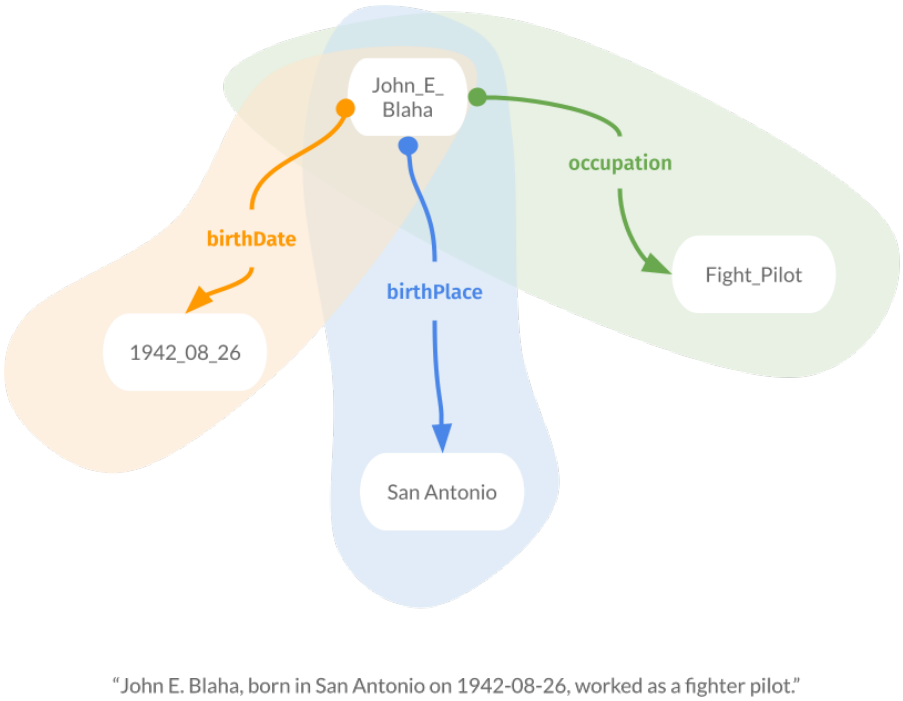


Figure 2.1 – **Example of data-to-text mapping.** The data is encoded by a set of RDF triples. In this particular example, each triple contain the same subject *John_E_Blaha*. The triples also contain the predicates *birthDate*, *birthPlace* and *occupation* and their associated objects *1942_08_26*, *San_Antonio* and *Fight_Pilot*. The goal of the rdf-to-text generation task is to map a set of RDF triples to a text. In this example, the set of triples is mapped to *John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot.*

This task can be seen as the automatic translation from sets of RDF triples into human language. Although the 2020 version of the challenge proposed to work on multilingual corpora, namely in English and Russian. We only studied the RDF-to-text task in English.

In this manuscript in general and in this chapter in particular, we call *RDF verbalization* and *lexicalization* the generation of a descriptive text given its corresponding RDF triples.

Indeed, the RDF-to-text task consists of mapping a set of triples to one of its possible lexicalization. It is to be noted that the generated verbalization must be syntactically and semantically correct. Moreover, it must contain the same level of information as the input data.

For the challenge, Gardent, Shimorina, et al. [2017a] released a public dataset. A first version of the dataset was introduced in 2017 with the challenge [Gardent, Shimorina, et al., 2017a]. An enhanced

2. https://webnlg-challenge.loria.fr/, last access: 09/02/2021

version of the same dataset was released with the 2020 version of the challenge [3]. We present the WebNLG Dataset in detail in the next subsection 2.1.3.

2.1.3 The WebNLG Dataset

The English WebNLG 2020 dataset contains RDF-to-text pairs divided into 16 *DBpedia* categories: Airport, Astronaut, Building, City, ComicsCharacter, Food, Monument, SportsTeam, University, WrittenWork, Athlete, Artist, CelestialBody, MeanOfTransportation, Politician, Company.

	Train	**Dev**
Entries	13.211	1.667
Lexicalizations	35.426	4.464
Distinct properties	372	290

Table 2.1 – **Size statistics of the WebNLG dataset.** Sizes of the official train/dev split of the WebNLG 2020 dataset.

For every input sample , several lexicalizations are provided. In Table 2.1, we provide some statistics on the train/dev split provided for the challenge.

The training set consists of $13,211$ unique triple sets associated with several possible lexicalization. A triple set contains up to seven RDF triples. Intuitively, a set containing more triples will be associated with more possible verbalization. We formed RDF-to-text pairs associating each possible lexicalization with its input triple set. This leads to a training set of $35,426$ samples. It is important to highlight that a lexicalization is not necessary a single sentence.

2.1.4 The Evaluation Protocol

To evaluate the participants in the challenge, the organisers conducted two test phases. The first phase was an automatic test phase using automatic evaluation metrics. Each participant could submit as many models as they wanted to be evaluated on these metrics.

Then, each team could submit one model for the second testing phase : the human evaluation.

First, we will have a look at the test data provided for the challenge and then we will focus on the evaluation protocol.

Test Data

The test set provided for the WebNLG challenge was split in three types of data that can be seen as three different tasks for our models.

3. https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/release_v3.0

Known entities and categories. The first split contains RDF triples that contain entities and categories seen in the training data. For instance the entity *Alan Bean* in the category *Astronaut*.

Unknown entities. The second split contains RDF triples with new entities within the same categories as the training data. For instance the category *Astronaut* is present in the train set but the entity *Yuri Gagarin* is never seen.

Unknown categories. The third split of the test set contain sets of RDF triples based on new categories not present in the training data.

Automatic metrics

The submitted models are evaluated using standard machine translation metrics: BLEU [Papineni et al., 2001], METEOR [Banerjee and Lavie, 2005], chrF++ [Popović, 2017], TER [Snover et al., 2006], the BERTscore [Zhang* et al., 2020] and BLEURT [Sellam et al., 2020].

As mentioned in Post, 2018, BLEU is a parameterized metric that may exhibit wild variations. The use of multiple metrics gives better outlooks on the general performance of the models.

BLEU (BiLingual Evaluation Understudy) BLEU is the most commonly used machine translation metric to evalute the quality of a generated text. The idea is to compare the generated sentence with one or several references generated by a human. By consequence, it is widely used for the evaluation of text generation tasks.

Introduced by Papineni et al. [2001], the idea is to compare the candidate text with one or several references by counting the number of n-grams that occur in both the reference and the candidate. BLEU is often computed for several values of n (most often $n \in \{1, 2, 3, 4\}$) and the scores are averaged geometrically. The BLEU score is in fact a modified n-gram precision score.

The BLEU score ranges from 0 to 1, values close to 1 representing more similar texts compared to the references.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) The METEOR metric was proposed and released by Lavie et al. [2004]. Banerjee and Lavie [2005] have then described the details underlying the metric.

The idea of METEOR is to compute a score based on a word-to-word alignement between the candidate and a reference. The word-mapping module features external NLP resources, namely a stemmer, a synonym lexicon and a paraphrase table in order to allow exact unigram matching to maching word stems, synonyms, and paraphrases.

After the initial alignement step, a parameterized harmonic mean of ungiram precision and recall is computed.

To take into account the word order, METEOR computes a penalty using a fragmentation fraction that compute the number of identical chunks in the candidate and the reference.

TER (Translation Edit Rate). Introduced by Snover et al. [2006], TER measures the amount of editing that a human would have to perform to change a candidate sentence so it exactly matches the reference. Possible edits include the insertion, deletion, and substitution of single words as well as shifts of word sequences. All edits have equal cost.

chrf++ (Character n-gram F-score) first introduced by Popović [2016] and then improved by Popović [2017] is an automatic metric based on character n-gram precision and recall, enhanced with word n-grams. The scorer calculates the F-score averaged on all character and word n-grams (the default character n-gram order is 6 and word n-gram order is 2). The arithmetic mean is used for n-gram averaging.

BERTscore Some previous metrics were surface metrics that directly compared the words or characters of the candidate sentence and references. There are other metrics that use embedding vectors. An embedding vector is a projection of a word or a sentence to a low-dimensional continuous space. Embeddings provide a semantic representation of the words or the sentence and can be leveraged to compute the semantic similarity of the candiate and the reference. Indeed in an embedding space, we assume that close vectors represent similar sentences.

Recently Devlin et al. [2018] introduced the contextual embeddings BERT (Bidirectional Encoder Representations from Transformers) that not just compute a mapping between a vocabulary and a continuous space but provide a contextualized embedding that will be different according to the other words of the sentence. When BERT was released, it achieved state-of-the-art results on several NLU (Natural Language Understanding) tasks.

Zhang* et al. [2020] introduced an automatic evaluation metric that uses BERT: the BERTscore. The BERTscore computes a similarity score for each token in the candidate sentence with each token in the reference sentence. The token similarity is a cosine between contextual embeddings. Then the scorer computes recall, precision and an F1 measure:

$$R_{BERT} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j \quad , \quad P_{BERT} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j \quad ,$$

$$F_{BERT} = \text{BERT}score = 2\frac{P_{BERT} \cdot R_{\text{BERT}}}{P_{BERT} + R_{\text{BERT}}} \quad .$$

BLEURT (Bilingual Evaluation Understudy with Representations from Transformers) [Sellam et al., 2020] is a learned evaluation metric based on BERT whose main idea is to model human judgments. As text generation models became more and more efficient, the automatic metrics like BLEU started to diverge from human judgments. To tackle this issue, [Sellam et al., 2020] proposes to pretrain a learned metric on millions of synthetic data and then fine-tune it on human judgments. The pre-training signals are automatic metrics like BLEU or the BERTscore. The fine-tuning is done on specific tasks. For the WebNLG challenge, the fine-tuning of BLEURT has been done on the three humanly evaluated aspects of WebNLG namely semantics, grammar, and fluency.

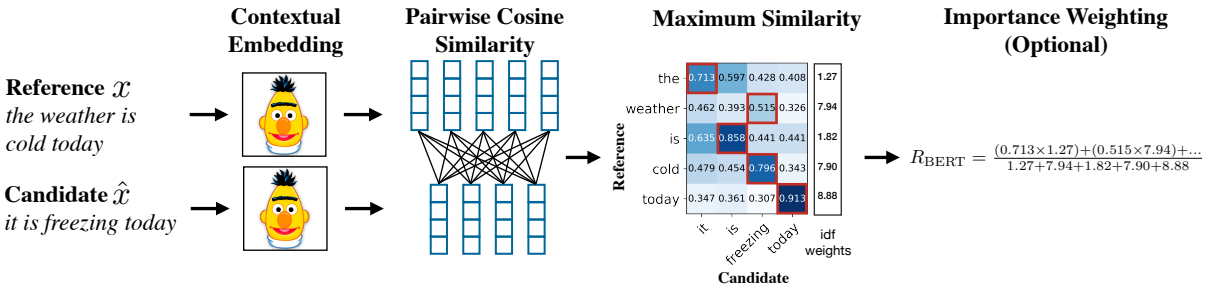


Figure 2.2 – Figure drawn from the original paper introducing the BERTscore [Zhang* et al., 2020]. The Figure illustrates the computation of the R_{BERT} score. The scorer computes the contextual BERTembeddings of the candidate and reference's words. Then the sentences are aligned using the pairwise cosine similarity. An optional idf importance weighting is then applied to compute the final R_{BERT} score.

We noticed that while BLEURT is supposed to correlate better with human judgments, it is however difficult to understand the sensitivity of the fined-tune BLEURT whitout the model itself.

2.1.5 Human Evaluation

For the RDF-to-text generation, the models outputs are assessed according to five criteria by native speakers. We report the criteria as they are presented in [Castro Ferreira et al., 2020]:

— Data Coverage: Does the text include descriptions of all predicates presented in the data ?
— Relevance: Does the text describe only predicates, which are found in the data ?
— Correctness: When describing predicates which are found in the data, does the text mention correct objects and aequately introduces the subject for this specific predicate ?
— Text Structure: Is the text grammatical, well-structured, written in acceptable English language ?
— Fluency: Is it possible to say that the text progresses naturally, forms a coherent whole and is easy to understand?

2.2 Background : the 2017 WebNLG Challenge Models

As presented in their report, Gardent, Shimorina, et al. [2017b] received eight models at the WebNLG 2017 challenge. They divided the submitted systems into three categories: pipeline systems, statistical machine translation (SMT) and neural machine translation (NMT). In the following subsections 2.2.1, 2.2.2 and 2.2.3, we present respectively each categories of models.

In this Section 2.2, we give an overview of the approaches used for the 2017 challenge, and we provide some required background elements.

The approaches used for the RDF-to-text task are standard approaches for text generation tasks such as machine translation or paraphrase generation.

In the part II of this thesis, similar approaches will be used for paraphrase generation and therefore we take advantage of this contribution to the WebNLG challenge to present the theoretical background of the work presented next.

2.2.1 Pipeline Systems

According to Gardent, Shimorina, et al. [2017b], three submitted systems used a template or grammar-based pipeline framework with some Natural Language Generation (NLG) module. This strategy follows the traditional NLG pipeline [Reiter and Dale, 1997] that focused on the creation of rules or templates to produce textual output.

Sun and Chris Mellish [2007] and Sun and Christopher Stuart Mellish [2006] have shown that most of the useful information for the generation is brought by the linguistic information than can be extracted from RDF predicates. In particular, they highlighted that the predicates can be divided into six categories based on their pattern. For instance, some predicates are a concatenation of a verb and a noun like *hasEmail*, or start with a verb and finishing with an adjective like *hasTimeOpen*. They developed handcrafted rules to construct linguistic patterns for each category. This technique enables a domain-independent verbalization since it is only based on the predicate pattern. On the other hand, Cimiano et al. [2013] used domain-dependent ontology lexicon to conduct a fine-grained and specific verbalization of the concepts. Although some methods tried to automatically learn those templates Duma and Klein, 2013, the main drawback of such approaches is the need of handcrafted rules and their poor capacity for generatlisation.

For the WebNLG 2017 challenge, three submitted models [Ferreira et al., 2017; Mille and Dasiopoulou, 2017; Phong and Dang, 2017] used this generation framework. Two models [Ferreira et al., 2017; Phong and Dang, 2017] relied on the extraction of rules or templates from the training data for surface realisation. The third model [Mille and Dasiopoulou, 2017] mostly focused on sentence planning with predicate-argument templates. For each properties in the trainind and testing data, they manually defined predicate-argument templates encoding many DBPedia features. In Figure 2.3, we report an example of predicate-argument template for the floor area of a building drawn from the model report [Mille and Dasiopoulou, 2017].

Figure 2.3 – Sample predicate-argument template for the floor of a building drawn from [Mille and Dasiopoulou, 2017].

The predicate-argument are then aggregated and rendered into sentences using rule-based graph transducers.

2.2.2 Statistical Machine Translation (SMT)

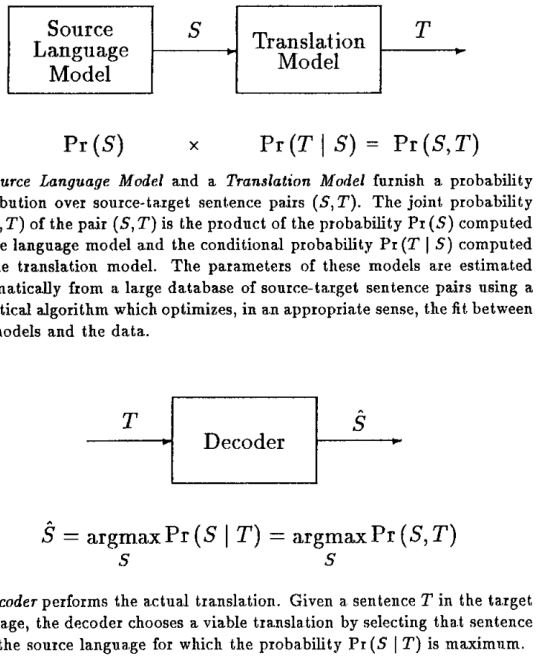


Figure 2.4 – **Statistical Machine Translation system.** Figure extracted from P. F. Brown et al. [1990]

Statistical machine translation was first introduced by Weaver et al. [1955] but developped later by P. Brown et al. [1988], P. E. Brown et al. [1993], and P. F. Brown et al. [1990].

In Figure 2.4, we report the illustration of a statistical machine translation system as depicted in P. F. Brown et al. [1990].

In the early 2000s phrase-based statistical machine translation (SMT) [Koehn, Och, et al., 2003] has been the dominant paradigm in machine translation research and de facto in many text generation tasks. In particular, the Moses open source toolkit Koehn, Hoang, et al., 2007 boosted the development of the topic.

For the WebNLG 2017 challenge, Ferreira et al. [2017] submitted a model which used the statistical machine translation framework. The model was trained on the WebNLG dataset using the Moses toolkit Koehn, Hoang, et al. [2007].

2.2.3 Neural Machine Translation : Supervised Sequence-to-sequence Neural Networks

Neural Machine Translation (NMT) is derived from phrase-based SMT. Its main divergences from SMT derive from the use of embeddings (continuous vector representations) for words or hidden stats.

There is no separate language model, translation and reordering models anymore but just one sequence model that predicts the tokens iteratively. NMT models uses artificial neural network.

In Chapter 1, we introduced in detail the sequence-to-sequence architecture and how it is trained and infered.

Supervised *Encoder-Decoder* models are at the core of neural machine translation.

2.3 Proposed Approaches for the 2020 WebNLG Challenge

As for mamy text generation tasks, KB verbalization lacks aligned data. In particular, it is difficult to exploit deep neural network architectures for these tasks because the number of parameters to be trained requires a large amount of data. In the Part II of this thesis, we present our work on paraphrase generation – another text generation task – on which we encoutered the same problem of lack of aligned data.

Hopefully, the dataset released by Gardent, Shimorina, et al. [2017a] for the challenge contains enough RDFs triples aligned to their corresponding lexicalizations to train neural models. For the 2017 version of the WebNLG challenge, the deep learning methods outperformed rule-based ones. At this time, deep learning methods relied mostly on attention-based *Seq2Seq* models [Bahdanau, Cho, et al., 2016; Sutskever et al., 2014] that are detailed in Chapter 1. The encoder-decoder models perform well but fall short of its generalization abilities when inferring unseen domains.

After the challenge in 2017, additional work has been published to advance the state of the art.

Y. Zhu et al. [2019] proposed to optimize the inverse KL divergence in order to generate higher-quality verbalizations. Trisedya et al. [2018] introduced a new neural network architecture based on Graph Neural Networks to keep the RDF structure in the encoding and manage better the RDF triples relationships. Following this work, several papers proposed graph-based encoder as a solution [H. Gao et al., 2020; Marcheggiani and Perez-Beltrachini, 2018; Moussallem et al., 2020; See et al., 2017; C. Zhao et al., 2020].

In parallel, Iso et al. [2020] introduced the *facteditor* model which applies multiple edits to turn an input sequence into the output sequence. They include it in a verbalization scheme which first generates a candidate verbalization that is then iteratively edited. The possibile editions of a word are *keeping* the word, *dropping* it or *generating* a new word.

Finally, based on the assumption that the RDF-to-text task is the inverse of the text-to-RDF task, J. Zhu et al. [2017], Tseng et al. [2020] and Guo et al. [2020] introduce a cycle training framework to learn both tasks simultaneously.

As concluded in Chapter 1, the recent advances in large pre-trained models and the Transformer architecture are very promising for text generation tasks. For the 2020 WebNLG Challenge, we proposed to explore the effect of massive pre-training and the Transformer model for the RDF-to-text task.

We chose to make use of external corpora during pre-training to reach a better generalization. We built two additional datasets for denoising pre-training and data augmentation purposes. This is motivated by the fact that participants' systems at previous WebNLG challenge suffered from high drops in

performance when evaluated on new entities and predicates not encountered during the training phase Gardent, Shimorina, et al., 2017b. Thus, by incorporating external corpora, we expect our system to outperform systems solely trained on the WebNLG dataset.

We give an extended introduction to the Transformer in Section 1.4. As highlighted in the original paper [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez, et al., 2017b], the individual attention heads from the multi-head attention layers learn to perform individually different tasks. In particular, they appear to learn the syntactic and semantic structure of the sentences independently. Mareček and Rosa [2019] has shown that the self-attention layers embed implicitly syntactic parse trees. Similarly, we hoped that our model would benefi from our pre-training stage by incorporating such syntactic information, while learning intra and inter-triples dependencies in the fine-tuning stage. We hoped that the Transformer model could learn the structures that were in the lexicalizations of the triples during the pre-training stage, and then learned the intra and inter-triples dependencies during the fine-tuning stage.

2.3.1 Data Augmentation

As mentioned earlier, the lack of data is the major obstacle to training large neural networks. In particular, we speculate that the poor generalization of the models on unseen categories is due to the lack of diversity in the training data. In the WebNLG corpus, there are only 747 unique subject entities and 385 unique predicates. Our goal was then to augment the training dataset with new RDF-to-text aligned data. For this purpose, we first collected a large corpus of sentences from which we then extraced RDF triples.

Sentence Collection

We used Wikipedia that offers a massive and free amount of data. The WebNLG dataset was constructed from DBPedia database which itself relies on Wikipedia.

The major advantage of including new data from Wikipedia is the integration of new named entities that are encountered from much more heterogeneous domains. We assume that this allows for a better generalization and knowledge integration. As observed in language models [Devlin et al., 2018; Radford et al., 2019; Z. Zhang et al., 2019], knowledge is somehow assimilated by the model throughout training.

We gathered 13,614 Wikipedia pages containing about 103 million sentences totally [4]. To alleviate the training process, we filtered sentences that may have obstructed the generation quality. We discarded a sentence (1) if it did not start with an upper case letter and did not end with a period, (2) if its length was longer than 50 or (3) if it contained special characters. After this first selection, 57 million unique sentences remained. We call this set of sentences WS1.

4. We used the *20200401* Wikipedia dump $\sim$ 18GB

Triple Extraction

For the step of triple extraction used the Stanford Open Information Extraction (Stanford OpenIE) tool [5]. Stanford OpenIE package is schema-free. That means that no preliminary definition of the possible predicates is required, as opposed to usual RDFs extractors. The raw text linking two entities will be retrieved as the predicate. Unfortunately, the returned triples may be incomplete, false or alike. For example, for the input sentence *"Barack Obama was born in Hawaii."*, the returned triples by Stanford OpenIE are ⟨*Barack Obama; was; born*⟩ and ⟨*Barack Obama; was born in; Hawaii*⟩. The first output triple is false and in some way expresses the same idea as the second one. Therefore, a filtering step is essential to reduce false triples. We developed simple rules to limit redundancy by comparing each extracted triple with another.

The filtering step worked as follow. First, we linearized each triple by concatenating words from the subject, predicate and object. To detect if the linearized version of the triples are equivalent, we computed the *Levenshtein distance* [Levenshtein, 1965] to consider minor variations between triples.

Two triples with an edit distance smaller or equal to 2 were considered to be similar. If triples similarity was confirmed, the longest triple was kept because sharing the more information with the input sentence. The higher the lexical coverage, the better in order to keep the sentence essential information. If input triples are considerably incomplete, the model may suffer from hallucination, *i.e.* it may generate content that is not present in the given input.

It is possible that the linearized version of the triples are too different in surface but semantically close. That is why more sophisticated conditions should be exploited. For this purpose, we made use of BLEU [Papineni et al., 2001]. Extracted triples are derived from the same sentence. Therefore, an *n-gram* based metric such as BLEU is a good choice to check if two triples are alike. We assumed triples to be analogous if their BLEU score was greater than 50. In such case, we kept the triple maximizing its BLEU score with the reference sentence for better coverage, as explained previously. The final collection of remaining triples-sentences pairs is called ST1.

Nevertheless, the detection of erroneous information conveyed by the triples is difficult. Aware that incorrect triples would directly compromise the performance of our model, we exploited this augmented data in a pre-training step, as detailed in Subsection 2.3.2.

2.3.2 Pre-training Strategies

Our proposal for the WebNLG 2020 challenge is to use Transformer models combined with a pre-training stage. In this section, we describe the latter.

As introduced in Section 1.4.2, NLP has gained from extensive pre-training strategies of word embedding models and language models. Recent pre-trained models like BART [Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Ves Stoyanov, et al., n.d.] exploit unlabeled data to boost model performance in a self-supervised setting. The denoising autoencoder objective from BART has shown a significant performance gain. A denoising objective aims to reconstruct a corrupted input. When fine-tuned on a text generation task, BART has confirmed to be highly effective. We therefore adopted a similar approach using our big Wikipedia sentences, *i.e.* WS1 dataset.

5. Available at https://nlp.stanford.edu/software/openie.html

Peters et al. [2019] use an arbitrary noising function to permute, delete, and mask words in the input. Our transformations differ from BART in that we required our corrupted input to contain factual information, similar to what our model would be exposed to when it was fine-tuned on our RDF-to-text task. With this in mind, we chose to keep words with specific Part-Of-Speech (POS) tags like nouns, verbs, adjectives and adverbs. Words with other tags were removed. Modal verbs (*e.g. should*) and passive forms were ignored as well. As an example, consider the Wikipedia sentence *"In 1860 few of the streets north of 42nd had been graded."*. After our noising transformations, we obtained the corrupted input *"1860 few streets north 42nd graded."*. We observe that remaining words stand for the *semantic mass* of the sentence from which we need to lexicalize and connect entailed concepts and entities properly.

Contrary to RDFs triples, the corrupted sentences have no inherent structure. In order to avoid too much divergence between pre-training and fine-tuning data, we followed the denoising pre-training with another pre-training stage on a RDF-to-text task. To do so, we used our constructed ST1 dataset which contains triple-based inputs, as detailed in Section 2.3.1. We expected that these two pre-training steps would improve our ability to generate faithful and consistent verbalization for the WebNLG Challenge.

We also tried without resetting the optimizer and obtained similar performances.

2.3.3 Curriculum Learning

In addition, we applied a Curriculum Learning (CL) approach for faster convergence and better local minima through the optimization process.

To recall, the idea of curriculum learning [Y. Bengio, Louradour, et al., 2009] is to gradually inscrease the complexity of the data while training to mimic the human learning behavior. In our case, we defined complexity as the number of RDF triples in the input. The higher the number of triples, the higher the complexity. We sorted the WebNLG pairs such that easier examples came first and then harder examples afterwards. Our curriculum approach differs from Y. Bengio, Louradour, et al. [2009] in that both easier and harder examples are seen within the same epoch, but in a gradually order. In their work, Y. Bengio, Louradour, et al. [2009] progressively add more complex samples during training.

To avoid shuffling our ordered training set for curriculum learning, we set the number of un-shuffled epochs to 30. From the 30 epochs onwards, batches were randomly selected. To showcase the effect of increasing complexity while training, we fine-tuned our pre-trained models with and without curriculum learning.

2.4 Experiments

We trained 8 Transformer models corresponding to 8 different training settings that we detail later in this section. Each model was then evaluated using the WebNLG Dataset test set.

2.4.1 Data Preprocessing

Moses Tokenization

For preprocessing, we used the Moses tokenizer [6] and subword segmentation following [Sennrich et al., 2016] with the subword-nmt library [7]. We left both the input triples and output text true-cased.

BPE Tokenization

In the context of the WebNLG challenge, we considered a transduction strategy as in [Gammerman et al., 1998a]. We restricted the vocabulary to the WebNLG dataset (training, validation and testing set). We forced our model to use a WebNLG-based vocabulary during pre-training so that the same model could be straightly fine-tuned without any vocabulary discrepancy. Technically, we loaded the last checkpoint from pre-trained model and directly started to fine-tune it by reseting the ADAM optimizer and setting the new data loader to the WebNLG training set.

This learning scheme aimed at performing well on a specific set, and not necessary to generalize. This is called transductive learning [Gammerman et al., 1998b].

The training process remained the same for all our experiments. Only the data loaded in the batches during training were changed.

To deal with the RDF triples format, we added to the vocabulary four special tokens, namely $\langle object \rangle$, $\langle subject \rangle$, $\langle predicate \rangle$ and $\langle eot \rangle$ (end of triple), that we used as separators within and between triples. In the case of multiple triples, we built the Transformer input sequence by concatenating triples one after the other. We used this input format on both ST1 and WebNLG.

For the denoising pre-training, we used the WS1 dataset. The transformed sentences (see Section 2.3.2) were directly fed to the Transformer. The model thus has to reconstruct the incomplete sentences. Examples of the preprocessed samples for the different training settings are reported in Table 2.2.

2.4.2 Training Settings

We used the Transformer implementation from FAIRSEQ library [Ott et al., 2019] [8] with the *transformer_base* hyper-parameters set defined by Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez, et al. [2017b]. We optimized the weights of our neural networks using an ADAM optimizer and a label-smoothed cross entropy loss.

We made 10 epochs of pre-training and stopped fine-tuning when the performance with the BLEU score on the validation set did not improved after 30 epochs. At the end, we kept the model that achieved the best BLEU on the validation set.

To study the impact of curriculum learning, we launched fine-tuning with and without it. In the former case, we prevented the shuffling of the batches for 30 epochs. The data being sorted by number of triples, the model had to deal first with simple samples, and then with more complex ones as the learning did progress. In the results, we report this setting as CL (for Curriculum Learning).

6. Available on github https://github.com/moses-smt/mosesdecoder
7. Available on github https://github.com/rsennrich/subword-nmt
8. We used the compiled version 0.9.0 from https://github.com/pytorch/fairseq

Task	Input sequence	Output sequence
WebNLG	⟨*subject*⟩ Italy ⟨*predicate*⟩ capital ⟨*object*⟩ Rome ⟨*eot*⟩	Rome is the capital of Italy .
	⟨*subject*⟩ Bionico ⟨*predicate*⟩ course ⟨*object*⟩ Dessert ⟨*eot*⟩ ⟨*subject*⟩ Bionico ⟨*predicate*⟩ ingredient ⟨*object*⟩ Raisin ⟨*eot*⟩	Bionico is a dessert which contains raisins .
WS1	He died he@@ art f@@ ail@@ ure h@@ ospital October 5 2014	He died of he@@ art f@@ ail@@ ure at the h@@ ospital on October 5 , 2014 .
ST1	⟨*subject*⟩ He ⟨*predicate*⟩ retired ⟨*object*⟩ 199@@ 0 ⟨*eot*⟩	He retired in 199@@ 0 .

Table 2.2 – **Preprocessed training samples.** The input RDF triples are concatenated into one sequence. We used the Moses tokenizer and a subwords encoding. The subwords are divided by @@. Special tokens are used to keep the RDFs triple structure : ⟨*object*⟩, ⟨*subject*⟩, ⟨*predicate*⟩ and ⟨*eot*⟩. In the case of the denoising task (*WS1*), we do not have these special tokens. We left the data truecased and no delexicalization is applied.

For decoding, we did a beam search with a beam of size 5. We merged the subwords back into words and detokenized.

2.4.3 Ablation Study Results

For evaluation, we used the official WebNLG evaluation script [9]. The metrics we used to compare our models are BLEU, METEOR, chrf++ and BLEURT. Relying on BERT's contextual embeddings, BLEURT offers semantically robust feedback. The *n-gram*-based evaluation techniques such as BLEU, METEOR or chrf++ are additional metrics to judge the generation quality. When used together, they give good assessment of the generation quality of our system.

Unlike in 2017, this time the generated sentences had to be detokenized and true-cased. Our models were therefore not directly comparable with the models of the previous version of the challenge.

Also, the test base being unavailable, we did not know how many references our generated sentences were compared to.

To assess the impact of the different parameters we wanted to explore, we conducted an ablation study. We defined our baseline as the Transformer trained only on the WebNLG dataset, without curriculum. We wanted to assess the influence of pre-training, data augmentation and curriculum learning compared to the baseline. To provide a fair and detailed analysis, we evaluated models on seen and unseen domains to shed light on the models' generalization ability. Results are given in Table 2.3.

9. Available on github https://github.com/WebNLG/GenerationEval

Seen categories							
WebNLG	WS1	ST1	CL	BLEU	METEOR	chrf++	BLEURT
✓				55.24	0.401	0.680	0.56
✓	✓			57.3	0.417	0.701	0.58
✓		✓		**59.32**	**0.428**	<u>0.712</u>	<u>0.6</u>
✓	✓	✓		57.49	0.420	0.702	0.59
✓			✓	54.78	0.399	0.676	0.56
✓	✓		✓	56.94	0.417	0.701	0.58
✓		✓	✓	58.36	0.422	0.703	**0.61**
✓	✓	✓	✓	<u>58.81</u>	<u>0.427</u>	**0.713**	<u>0.6</u>
Unseen entities							
WebNLG	WS1	ST1	CL	BLEU	METEOR	chrf++	BLEURT
✓				12.9	0.167	0.319	-0.62
✓	✓			29.16	0.301	0.518	0.1
✓		✓		**35.77**	**0.326**	**0.565**	**0.26**
✓	✓	✓		32.33	0.310	0.535	0.18
✓			✓	11.94	0.156	0.295	-0.68
✓	✓		✓	28.83	0.295	0.509	0.05
✓		✓	✓	31.99	0.312	0.537	0.16
✓	✓	✓	✓	<u>32.69</u>	<u>0.315</u>	<u>0.542</u>	<u>0.19</u>
Unseen categories							
WebNLG	WS1	ST1	CL	BLEU	METEOR	chrf++	BLEURT
✓				11.17	0.162	0.310	-0.64
✓	✓			21.02	0.265	0.452	-0.06
✓		✓		<u>23.26</u>	**0.288**	**0.485**	**0.03**
✓	✓	✓		**23.42**	0.275	<u>0.469</u>	<u>-0.02</u>
✓			✓	12.45	0.156	0.298	-0.63
✓	✓		✓	21.84	0.263	0.451	-0.04
✓		✓	✓	22.84	0.273	0.465	-0.05
✓	✓	✓	✓	22.72	<u>0.276</u>	0.466	<u>-0.02</u>

Table 2.3 – **Experiments results**. Automatic evaluation on the official WebNLG test set. For each learning strategy, we provide performances on seen categories (top), unseen entities (middle) and unseen categories (bottom). Bold and underlined values correspond to the best and second-best results respectively.

In Table 2.3, when comparing our baseline with pre-training strategies for each category (no curriculum), we note an average rise of 3.07, 19.6 and 10.97 respectively in BLEU with pre-training. Similar variations can be noted with METEOR, chrf++ and BLEURT metrics, albeit less striking. Based on BERT's contextual embeddings, BLEURT gives a good estimate of the semantic correlation between prediction and references. Most of the time, *n-gram*-based and semantic metrics show perfect harmony. Top values of BLEURT are obtained for the same models as *n-gram*-based metrics. Therefore, all metrics tend to correlate proving a good agreement. The high improvements over unseen domains are easily explained due to the diversity of augmented data. New entities and domain-specific lexicon encountered better help to model out-of-distribution data relations. Thus, such results underline usefulness of external corpora and strengthen the need of pre-trained model to lexicalize KBs.

On seen categories, our baseline gives a BLEU score of 55.24. However, for out-of-domain generation, all models demonstrate severe shortcomings. Tested on unseen entities, our baseline shows a BLEU drop of 42.34 to reach 12.9. We witness similar and even more important loss in unseen categories. In the case where predicates are unknown to the model, it is hard to generate a consistent description of the input RDFs. From seen categories to unseen categories, our baseline is nearly penalized by a factor of 5. Such effect is tempered with our pre-training approaches. The average drop in BLEU of pre-trained models (without curriculum learning) is 25.6 from seen categories to unseen entities, and 35.5 from seen to unseen categories.

Unexpectedly, when a curriculum learning approach is used, we witness drops in performance. This is counter-intuitive and opposed to previous experiment on our validation set. CL seems to help lightly when the model was pre-trained with both external corpora WS1 and ST1. This result may highlight an overfitting issue.

It is interesting to note that the best results are revealed with a pre-training on ST1, exclusively. With 5 times less data, ST1 leads to better performance. The extracted triples surely include inaccurate triples. Despite the imperfect quality of the ST1 dataset, its use contributes to generalization ability.

On the contrary, we report that denoising pre-training does not show satisfying results when combined with our pre-training on the ST1 dataset, leading eventually to a negative effect. This may be caused by an input distribution discrepancy between WS1 and ST1 input. Denoising pre-training doesn't require triples as input but a noisy sentence. Mismatch between this representation and the triples linearization may be the reason of such side-effect.

As for the WebNLG 2020 challenge, participants were requested to submit a single model for evaluation. In our case, we decided to submit our Transformer pre-trained with both WS1 and ST1 (without curriculum). After an analysis of the generated text, the model trained with much more data has a tendency to be much more fluent and to aggregate information in a better way. Castro-Ferreira et al. [2020] report human evaluation based on different criteria: data coverage, relevance, correctness, text structure and fluency. For each criterion, a value between 0 (complete disagreement) and 100 (complete agreement) is given. The scores are normalized and then clustered into groups such that models of a same cluster do not show any significant statistical differences in their scores [10]. When tested on seen categories, we note that our submitted model is competitive with other teams' models. However, on unseen data,

10. Details of the evaluation procedure is outlined in Castro-Ferreira et al., 2020 and scores are publicly available at https://gerbil-nlg.dice-research.org/gerbil/webnlg2020resultshumaneval

although a significant improvement over a simple training of a Transformer, our model shows limitations compared to other participants. A lower rank is systematically observed for both unseen entities and unseen categories. We assume that a delexicalization step and a much massive pre-training as in [Devlin et al., 2018; Radford et al., 2019] would help to improve our generalization ability on unseen domains.

Part II

The Use Case of Paraphrase Generation

Chapter 3

The Paraphrase Generation Task

Paraphrase generation is a fundamental task in NLP.

Paraphrasing - or the act of rephrasing - a sentence or piece of text is used in several real-life contexts. Writers often reword their paragraphs to be more concise or to adapt the vocabulary used for a specific audience. It is also widely used in journalism where several articles convey the same information often derived from the same source. Examples can be found every day; like in June 2021 after a football match, two British newspapers headlined their articles as follows: **"England beat Germany as Sterling and Kane send them to Euro 2020 last eight"** [1] and **"England beat Germany as Raheem Sterling and Harry Kane score to reach Euro 2020 quarters"** [2]. These two headlines are paraphases of each other.

Paraphrase generation can be roughly considered as a variant of machine translation where the translation is performed in the same language as the input. As a consequence, the advances in paraphrase generation are tighly linked to the advances in machine translation. Paraphrase is however much less studied. The main reason is probably the lack of resources available. In particular, state-of-the art models for machine translation are supervised neural networks trained on huge aligned datasets. Such datasets are not available for paraphrase generation.

Another key difference between machine translation and paraphrasing is that generating paraphrases requires by essence more diversity than machine translation: a simple copy of a sentence is obviously not a good paraphrase.

In the next section 3.1, we will formalize what a paraphrase generator is and we will provide a few examples of its NLP applications.

In Section 3.2, we will describe the corpora of aligned or comparable paraphrases that we use in the paraphrase generation experiments of Chapters 4 and 5. Finally, in Section 3.3, we will detail the protocols and metrics that we used to evaluate the performances of the paraphrase generator models.

3.1 Definition and Applications

Automatic paraphasing has been an active research topic for some time now. Paraphrase generation is a task that is often approached as a componant of other tasks. McKeown [1979] developped a paraphraser

1. David Hytner, **The Guardian**, url: https://www.theguardian.com/football/2021/jun/29/england-germany-euro-2020-last-16-match-report, last access : 07/01/2021

2. Sam Wallace, Alan Tyers and Uche Amako, **The Telegraph**, url: https://www.telegraph.co.uk/euro-2021/2021/06/29/england-vs-germany-live-euro-2020-score-team-news-updates/, last access : 07/01/2021

component for a question-answering system. Meteer and Shaked [1988] combined a natural language understanding (NLU) system with a natural language generation (NLG) system to build a paraphrase generator model. They also investigated the role of paraphrasing in a cooperative dialog system.

In fact, paraphrase generators are useful for many downstream tasks. The underlying task is decisive with the available resources for the paraphrase generator development. In Subsection 3.1.2, we give an overview of the applications of paraphrase generation.

The task of paraphrase generation is difficult due to the structure of the langage itself. Both McKeown [1979] and Meteer and Shaked [1988] highlighted the issue of ambiguity when paraphrasing. An example given by McKeown [1979] is the following question : *Which students read books on computers dating from the '60s ?*. This sentence has two equally valid interpretations; either the computers are dating from the '60s or the books. Both possible interpretations lead to different paraphrases that could be either *Assuming that there are books on computers (those computers date from the '60s), which students read those books ?* or *Assuming that there are books from the '60s on computers, which students read those books ?*

The concept of paraphrase that we formally define in subsection 3.1.1 is simple but it leads to several subtilities that need to be clarified.

3.1.1 Definition

We give here a definition of a **paraphrase** drawn from an English dictionary.

Paraphrase

As defined in the online Oxford University Press dictionary[a]:

- *verb* : Express the meaning of (something written or spoken) using different words, especially to achieve greater clarity.
- *noun* : A rewording of something written or spoken.

a. (2021). Definition of paraphrase [online]. Oxford University Press. Available at: https://www.lexico.com/definition/paraphrase (Accessed: 2 July 2021).

The noun « *paraphrase* » itself is ambiguous. In this thesis, we will only talk about paraphrase pairs that we define next. The term paraphrase can be used at different levels of text granularity. One may want to paraphrase a full text, a sentence, a phrase or only a word.

Paraphrase pair

As defined by Dras [1999]:

A **paraphrase pair** is a pair of units text deemed to be interchangeable.

In Table 3.1, we give examples of pairs of paraphrases drawn from aligned corpora. Paraphrase pairs can be used for two distinct tasks: paraphrase identification and paraphrase generation.

Paraphrase identification task

The **paraphrase identificaton task** consists of a binary classification task. Given a pair of sentences as input, a paraphrase identifier outputs if yes or no the input sentences are paraphrases of each other.

Paraphrase generation task

The **paraphrase generation task** consists of a text generation task. Given an input sentence, a paraphrase generator outputs its paraphrase.

We highlight here that the paraphrase generation task is more difficult than the paraphrase identification task because a paraphrase generator needs to first generate a valid sentence and second generate a sentence semantically similar to the input one. In Section 3.3.2, we report an experiment on paraphrase identification. Otherwise, all the experiments presented in this Part of the thesis involve paraphrase generation.

A paraphrase generator is useful in many applications.

3.1.2 Applications

Paraphrase generation is a fundamental task of NLP. Indeed, the capacity of being able to generate paraphrases of sentences to provide diversity and coverage finds applications in several domains like machine translation, question answering, improving evaluation metrics, dialog systems, or adversarial learning. In the following paragraphs, we give a quick overview of a few applications of paraphrasing to give an insight into the value of an efficient paraphrase generator.

Question-Answering and Dialogue Systems S. Gao et al. [2020] proposes a framework that jointly trains a paraphrase model and a response generation model to improve a global dialogue generation model. For that, they constructed a high-quality dialogue paraphrase corpus which is used to train a paraphrase generation model. Their experiments show that their paraphrase-augmented model does not suffer when reducing the training data and is still efficient in low resource settings.

Gardent and Barahona [2013] explores how paraphrasing can increase the accuracy and the robustness of supervised models in dialogue systems. They argue that training corpora for question answering (QA) characters should be large, balanced and varied. They explored the different ways of augmenting the training data of a supervised QA engine. In order to do that, they expanded the size and quality of the training corpus using paraphrase generation techniques.

Information extraction and retrieval

Information retrieval (IR)

Finding documents of an unstructured nature (text) that satisfies an information need from within large collections, using queries. Often, IR systems compute a similarity score on how well each documents in the database matches the query, and rank the documents with respect to this score.

Culicover [1968] explores the use of paraphrases to effect the retrieval of stored English texts and the information contained in stored English texts. The idea is to add a paraphrasing module in the IR pipeline. Roughly, the IR pipeline select keyphrases and keywords of an English text and try to match the extracted keywords to another keyword of the target area in the text. If a match is found, the model returns the text surrounding the keyword. This is called the kerword-keyphrase method. To make this pipleline more robust, Culicover [1968] added a paraphrasing module that automatically generates a list of paraphrases of the keyword to be matched with the text, improving the chances of matching and retrieving paraphrased informations.

A lot of research has been conducted to explore paraphrase generation for generating similar or related queries since. [Beeferman and Berger, 2000; R. Jones et al., 2006; Metzler et al., 2007; Sahami and Heilman, 2006; Shi and C. C. Yang, 2007]

Information extraction (IE)

Task of automaticaly extracting structured information from unstructured documents. For example, knowledge base population is an IE task. It consists of filling a database of facts stored as RDFs triples given a set of documents.

Sekine [2006] introduces an IE system that extracts the information with respect to the user's query by creating patterns and building tables using paraphrase discovery. The paraphrase matching module finds patterns that lead to the same table.

Improving machine translation Paraphrasing has been used to improve machine translation following two paths. Firstly, it can be applied to improve the translation process itself. Secondly, paraphrasing is useful in the evaluation of machine translation systems.

Callison-Burch, Koehn, et al. [2006b] used paraphrases to improve an SMT system. To be more precise, they allow any source phrase to be translated by the translation of its paraphrase. In other words, any unknown source phrase can be paraphrased before being translated. They show that this strategy achieves an improvement in coverage and translation quality.

Evaluating machine translation systems is a challenging problem due to the high quality of the recent machine translation models. Freitag et al. [2020] aims to show that the metric is important but the nature of the references is also important. In particular, some references have poor diversity and lack coverage.

Recently, paraphrase generation has been used to create more robust automatic metrics for machine translation. Indeed, automatic metric scores compare the generated sentence with one or several reference sentences. But in some cases, the model would generate valid sentences for the task but very different from the reference sentence used for evaluation. The automatic metrics would record bad performances while a human annotator would accept the generated sentence. That is why the automatic metric scores correlate badly with human judgment.

For instance, the top systems for English to German translation are different with respect to automated or human evaluation [Barrault et al., 2019; Bojar et al., 2018].

Sellam et al. [2020] introduced a learned metric BLEURT for text generation task and in particular machine translation to try to tackle this issue. To train BLEURT, they used existing sentence pairs datasets and added automatically generated sentence pairs using three techniques. One of them is using a backtranslation strategy that consists of translating from English to another language and then back to English with a translation model. This paraphrasing model can produce mispredictions but BLEURT uses them as a source of realistic alterations in order to be able to identify them.

Iyyer et al. [2018] introduced *syntactically controlled paraphrase networks* (SCPNs) that are trained to generate paraphrase with respect to a desired syntax. This piece of work highlights that SCPNs generate paraphrases at least as good as uncontrolled paraphrase systems would but with the ability to generate syntactically adversarial examples. Such adversarial examples can fool pretrained models and be used in a data augmentation scheme to improve the models robustness to syntactic variation.

I. J. Goodfellow et al. [2014] showed that many learned models can be fooled by adversarial examples. Such adversarial examples can be easily generated manually by introducing lexical or syntactic variation not seen in the training set.

This non-exhaustive list of potential applications of paraphrase generation shows that paraphases are useful in several NLP applications and as a consequence a very active reseach topic.

3.2 Data

There exists no generic aligned paraphrases dataset. The available aligned paraphrase corpora are often biased toward specific problems like *question answering* or *image captioning*. In this section, we introduce the different paraphrase corpora divided in two categories according to how they were created. First, there are the paraphrase identification corpora that have been crafted for paraphrase identification specificaly. Secondly, there are two other corpora that have been built following another objective namely *image captioning* and *subtitles translation* but can be seen as paraphrase corpora.

3.2.1 Aligned Corpora Built for Paraphase Identification

MSRParaphrase

At the opposite of Statistical and later Neural Machine Translation, research works on paraphrase generation and identification lacked large-scale labeled corpora of sentential paraphrases for a long time.

In an attempt to fill this gap, Dolan and Brockett [2005] released the Microsoft Research Paraphrase Corpus (MSRParaphrase).

The MSRParaphrase dataset consists of sentence pairs initially extracted from the Web. An initial database was first constructed with two heuristics based on shared lexical properties and sentence position in documents. A second database was then distilled from the first one with more precise criteria on the size of the sentences and the number of words overlap between the two sentences. Finally the sentence pairs were automatically classified as paraphrases or not with a low threshold set to deliberately over-assume paraphrase pairs. This strategy allows to extract paraphrase pairs and "likely" paraphrase pairs. This last updated database was then humanly annotated and split between paraphrase and non-paraphrase pairs.

The MSRParaphrase dataset contains mostly long sentences drawn from pieces of news. It is a small but high-quality paraphrase identification corpus that was labeled by humans.

Here is a couple of **paraphrase** pairs drawn from MSRParaphrase:

- `If people took the pill daily, they would lower their risk of heart attack by 88 percent and of stroke by 80 percent, the scientists claim.`
- `Taking the pill would lower the risk of heart attack by 88 percent and of stroke by 80 percent, the scientists said.`

- `Retail industry experts predict the next five days will likely make or break Christmas 2003 for many retailers.`
- `As the holiday shopping season peaks, industry experts predict the coming week could make or break Christmas 2003 for many retailers.`

Here is a couple of **non-paraphrase** pairs drawn from MSRParaphrase:

- `A tropical storm rapidly developed in the Gulf of Mexico Sunday and was expected to hit somewhere along the Texas or Louisiana coasts by Monday night.`
- `A tropical storm rapidly developed in the Gulf of Mexico on Sunday and could have hurricane-force winds when it hits land somewhere along the Louisiana coast Monday night.`

- `Both studies are published in the Journal of the American Medical Association.`
- `The study appears in the latest issue of the Journal of the American Medical Association.`

QQP

Quora Question Pairs (QQP or Quora) is a dataset [Kornél Csernai, 2017] is a paraphrase identification corpus dedicated to question-answering systems.

The main purpose of the QQP dataset was to improve the duplicate questions identifier for the Quora website [3] which is a question answering website. Quora's users can post questions to be answered by other users. Quora tries to keep each question page associated with a unique question and avoid duplicate question pages.

To tackle the issue of questions duplicates, Quora released a tool called *question merging* [4] that allows a user to merge several duplicate questions at once. This tool is available to any user from any question page. The merges are then approved or not by bots. Using this duplicate tagging tool, Quora built a dataset of duplicate questions that can be considered paraphrase pairs.

In order to build a duplicate identification dataset, they added negative examples using the *related questions* associated with each questions on Quora. The *related questions* are then on similar topics but not semantically equivalent.

QQP is then an ideal paraphrase identification corpus with paraphrase pairs and non-paraphrase but still close sentences pairs. The main drawback of QQP is its bias towards question answering. Indeed, all the sentences that belong to QQP are questions. On top of that, the data annotation process using *question merging* is noisy and the labels are not guaranteed to be perfect [Kornél Csernai, 2017].

We provide examples of pairs of sentences drawn from QQP in the first part of Table 3.1.

PAWS

The $\text{PAWS}_{\text{wiki}}$ dataset [Y. Zhang et al., 2019] is a recent paraphrase identification corpus that contains several lexicaly similar but hard-to-classify pairs like *"Flights to Florida from New-York"* and *"Flights from Florida to New-York"*. The PAWS corpus was introduced to adress the lack of sentence pairs with high-lexical overlap that are being paraphrases.

Y. Zhang et al. [2019] generated challenging pairs using word swapping and back translation. The PAWSdataset is divided into two categories: PAWS_{QQP} generated from the QQP dataset, and $\text{PAWS}_{\text{Wiki}}$ generated from Wikipedia. As we already used the QQP datasets using our own train/dev/test split; we only considered the $\text{PAWS}_{\text{wiki}}$ for our experiments. In this manuscript we will use PAWS and $\text{PAWS}_{\text{wiki}}$ equivalently, and whenever we mention the PAWScorpus, we will refer to the $\text{PAWS}_{\text{wiki}}$ part only.

We provide examples of pairs of sentences drawn from PAWS in the second part of Table 3.1.

3.2.2 Comparable Sentences Corpora

MSCOCO

The *Microsoft Common Objects in Context* (MSCOCO) dataset [Lin et al., 2014] was first built as an scene understanding and object recognition dataset. The dataset gathers images of common objects in

3. www.quora.com
4. https://quorablog.quora.com/Introducing-Question-Merging

Sentence 1	**Sentence 2**	**Label**
Why are African-Americans so beautiful?	Why are hispanics so beautiful?	0
Is there a reason why we should travel alone?	What are some reasons to travel alone?	1
What was the deadliest battle in history?	What was the bloodiest battle in history?	1
How is vanilla extract made?	How do you make sugar cookies without vanilla extract?	0
are aliens real or are they fake ?	Do aliens exists?	1
The Tabaci River is a tributary of the River Leurda in Romania .	The Leurda River is a tributary of the River Tabaci in Romania .	0
The Tabaci River is a tributary of the Leurda River in Romania .	The river Tabaci is a tributary of the River Leurda in Romania .	1
It was that Easipower said ,	It was Easipower that said :	1
The tracks were produced by Tommy Lee , and feature Michael Beinhorn on drums .	The tracks were produced by Michael Beinhorn and have Tommy Lee on drums .	0
The episode was written by Chuck Tatham and led by Fred Savage .	The episode was written by Fred Savage and directed by Chuck Tatham .	0

Table 3.1 – Examples of pairs of paraphrases drawn from the corpora QQP [Kornél Csernai, 2017] and PAWS [Y. Zhang et al., 2019]

natural contexts. The images were then annotated according to different tasks: object category labeling, object instance spotting, image segmentation, and image captionning.

We focus here on the latter: image captionning. With each image in the dataset, they added several captions. The description of the image captionning annotation is provided on a separate publication [Chen et al., 2015].

The captions were provided by human annotators using Amazon's Mechanical Turk (AMT). Several annotation instructions were given; for instance, it was asked to describe all the important parts of the scene and not to focus on unimportant details; not to interpret the past or future of the moment captured by the image nor describing what a person might say or her name. There was also a constraint on the size of the sentence.

Assuming that two captions of one image are semantically close to each other as they describe the same picture, the MSCOCO dataset can be seen as a paraphrase dataset. We assume that two captions of the same image are paraphrases.

The MSCOCO dataset is widely used in the paraphase community for two main reasons. First, the corpus is big. Second, the paraphrases come as set of several captions instead of just pairs. The different possible paraphrases can be used at test time to compare with a candidate paraphrase which provide coverage and diversity, especially for surface-based metrics like BLEU. However, in practice, the bias over image description is strong and the quality of the paraphrases is often questionable.

On Figure 3.1, we display images extracted from the MSCOCO dataset associated with some of their respective captions. We can notice the limits of such corpora for paraphrase generation. Some captions lack elements mentionned in others. On the third image in the figure, the annotators do not agree on the nature of the bear (*The large bear is made up of clay.* and *A teddy bear that appears to be made out of wood.*). As a consequence, the captions are not semantically equivalent.

The first MSCOCO dataset was released in 2014. In 2017, they released a new version with more annotations. In this manuscript, when we write about the MSCOCO dataset, we refer to the 2017 version.

OPUSPARCUS

The OpenSubtitles Paraphrase Corpus (OPUSPARCUS) dataset [Creutz, 2018] is a corpus of paraphrases extracted from the OpenSubtitles2016 corpus [Lison and Tiedemann, 2016] which contains subtitles from movies and TV shows.

The strategy to build the paraphrase corpora is to use the *pivot language* technique introduced by Bannard and Callison-Burch [2005]. The technique consists in translating a source sentence into another language - the pivot language - and translating back. That way, the last translation is a paraphrase of the initial source sentence.

The OPUSPARCUS corpus is interesting for the nature of the data that are more informal and colloquial sentences. However, according to the authors, in the data there are many sentence pairs that only slightly differ from each other. This lack of diversity between the paraphrases may be problematic to train a paraphrase generator.

Here is a few **paraphrase** pairs drawn from OPUSPARCUS:

- `When 'd you last see him ?`
- `When was the last time you saw him ?`

- Teddy bear next to monkey in foreground of picture.
- A stuffed teddy bear and monkey share the picture.
- two stuffed animals a bear and a monkey

- Two cups of fruits placed next to each other.
- Pair of fruit cups on kitchen marble counter.

- A carved bear that has a ribbon around the neck.
- A wood carved bear is sitting on a table.
- a statue of a bear wearing a red ribbon
- A teddy bear that appears to be made out of wood.
- The large bear is made up of clay.

Figure 3.1 – Sample pictures from the MSCOCO dataset associated with captions written by human annotators. Several captions associated with the same image can be considered as paraphrases. In practice, we notice that not all of them are necessarily semantically close. For example, for the third image, one annotator speaks of a clay bear where another see a wooden bear. Also, a caption may focus on more or less elements in the picure. For instance, for the second image, the second caption mention the *marble counter* that is not mentionned in the first caption.

- `Oh , what year is this ?`
- `You did all you could do .`
- `It 's gonna be okay .`
- `What year is it , anyway ?`
- `You did everything you could .`
- `It 's all going to be fine .`

3.2.3 Statistics on the Aligned Corpora

Each corpus has been split into three set : *train*, *dev* and *test*. In this Section, we provide a description of the split process for each corpora. In Table 3.2, we provide basic statistics on the different aligned corpora. In particular, we give the exact size of the *train/dev/test* sets and the median number of words in the sentences contains in each corpus.

MSRParaphrase

The original MSRParaphrase dataset comes with a *train/test* split of aligned sentence pairs. We first filterd out the pairs labeled as non-paraphrase. Then we sampled 500 pairs from the *train* split to make out *dev* split.

QQP

The original QQP dataset contains 149k pairs of sentences annotated as same questions. We assume these are paraphrases. We sampled 4k to make a *test* set. Then we sampled 145k data as the *train* set and the rest as *dev* set. In some papers [Gupta et al., 2018] released before 2017, authors created several version of the train set by sampling 150k, 100k or 50k data. However, the official QQP release was cleaned in 2017, and the number of samples went from 155k to 149k. Also, to be consistent with the methodolgy of other publications, we generated a bidirectional version of the dataset by swapping the source and target sentences and doubling the number of samples in the training set.

PAWS

We only kept the PAWS_{Wiki} data. The original data comes as "labeled" or "unlabeled". The "labeled" data have human judgements while the "unlabeled" data have noisy labels without human judgements. All data are generated from both word swapping and back translation methods. We kept the original *train/dev/test* labeled split and added the noisy data as an auxiliary training set.

MSCOCO

The original data were split as *train* and *valid* splits. Also each image was associated with 5 captions. We kept the original train split as our train split to be consistent with the litterature. In order to obtain an aligned paraphrases corpus, we created every paraphrase pairs from the captions sets in the train split. For the valid split, we kept one sentence as the source sentence and the other captions as references. That way, we have several references to compare with the generated candidate. We sampled 5000 samples of the *valid* set as our *test* split and the remaining samples as our *dev* split.

OPUSPARCUS

For OPUSPARCUS , we kept the original *train/test/dev* split. The *train* set is automaticaly annotated while

the other two are manually annotated by two different annotators. Each sentences pairs is associated with a score that is a combination of the two annotators score. In the annotating scale, a mark of 4 means that the two sentences are paraphrases, 3 means that they are almost paraphrases. A score of 3.5 for a pair of annotators means that one annotator scored the pair with 4 and the other annotator with 3. In our final aligned paraphrases dataset, we only kept the sentences pairs that had a score (automatic or manual) of at least 3.5 that corresponds to paraphrases pairs.

In Table 3.2, we give the median number of words in the sentences contains in each corpus.

We can note that MSRPARAPHRASE and PAWS contain mostly long sentences while OPUSPARCUS short ones. Also, these statistics highlight the difference in size of the different corpora. The big size of MSCOCO explains its wide utilisation in the paraphrase community especially for neural network training.

Corpus		MSRPAR.	QQP	PAWS	MSCOCO	OPUSPAR.
Nb. of samples	train	5 006	290 000	695 170	2 369 206	919 872
	dev	500	526	17 078	20 000	1 456
	test	2 294	4 000	7 072	5 000	1 446
Median nb. of words		23	10	20	11	6
Nature bias		News articles	Questions	Lexical overlap	Image captioning	Dialogues

Table 3.2 – Basic statistics on the aligned paraphrase corpora. Each corpus has been split into train, dev and test sets. We display the number of samples in each split for each corpus. We also provide the median number of words in the sentences of each corpus. One can notice that MSRPARAPHRASE has mostly long sentences while OPUSPARCUS short ones. We also recall the biases associated with each corpus in the last row of the table.

Also, it is worth mentioning that for MSCOCO and QQP , that are the two mostly used corpus for evaluation paraphrase generators, there are no official *test* set. While there is an overall consensus on the size of the *test* set, each publication records performances on a different sampled test set.

3.3 Evaluation

The automatic evaluation metrics are similar to the one used for the WebNLG Challenge presented in Subsection 2.1.4 namely BLEU, METEOR, TERand the BERTscore. As we are willing to make the two parts of this thesis independantly readable, we recall in the following Subsection 3.3.1 the definitions of the metrics already defined in Subsection 2.1.4. We add other metrics used in the paraphrase generation experiments. In Subsection 3.3.2, we present a little experiment on paraphrase identification carried on QQP and PAWS using the automatic metrics BLEU and BERT score as classifiers.

3.3.1 Automatic Metrics

Automatic evaluation of text generation requires a comparison of candidate sentences to annotated references. One can use automatic metrics with aligned corpora presented in Section 3.2 using the pairs of paraphrases as source and reference pairs. We recap here the automatric metrics. Some were already defined in Section 2.1.4 for the WebNLG Challenge.

BLEU

BLEU is the most commonly used machine translation metric and by consequence very used in paraphrase translation. Introduced by Papineni et al. [2001], BLEU counts n-gram overlap between the candidate and the references. The idea is to compare the candidate with one or several references by counting the number of n-grams that occur in both the reference and the candidate. BLEU is often computed for several values of n (most often 1,2,3,4) and the scores are averaged geometrically. BLEU is a surface metric and if the candidate sentence and the references do not use the same vocabulary, they will obtain a very low BLEU score. This is what we want when the candidate is bad. However the candidate may contain synonyms or phrasal paraphrases of words in the reference and be completly valid. Such candidate would obtain a low BLEU score.

METEOR

Banerjee and Lavie [2005] tackles this issue by introducing METEOR. METEOR uses external resources, namely a stemmer, a synonym lexicon and a paraphrase table, in order to allow exact unigram matching to machting word stems, synonyms, and paraphrases.

On can use other metrics designed for more specific tasks.

Levenshtein Distance

Word edit distance or word error rate [Levenshtein, 1965] computes the number of edit operations required to get from the candidate sentence to the reference.

TER

TER [Snover et al., 2006] is an enhanced version of the edit distance by normalizing it by the number of reference words. In other words, TER measures the amount of editing that a human would have to perform to change a candidate sentence so it exactly matches the reference.

Embedding vectors metrics

At the opposite of the surface metrics, the embedding metrics uses sentence or word embeddings information to compute the semantic similarity of the candidate and the reference. One can leverages the BERT contextual embeddings [Devlin et al., 2018] for instance.

BERT score

We call BERT score the F_{BERT} metric introduced by Tianyi [2019]. The BERT score leverages the contextual embeddings BERT introduced by Devlin et al. [2018] that provide a contextualized embedding of the tokens in a sentence. The BERT score computes a similarity score for each token in the candidate paraphrase with each token in the reference sentence. From these scores, the scorer then computes an recall, precision and an F1 measure.

Scoring the datasets

The task of generating paraphrases is special in the way that the source sentence is a paraphrase of the target sentence. Unlike the translation task where a transformation must be performed on the source sentence to obtain a candidate translation, for the paraphrase task the source sentence is already a candidate paraphrase.

The aligned paraphrase datasets presented in Section 3.2 contains paraphrases by definition. We can therefore evaluate our automatic metrics on the datasets directly.

We computed the BERTscore and the Levenshtein distance between the two sentences of each paraphrase pairs for all corpora. We report in Table 3.3 the rough distribution of the two scores on the corpora.

Corpus	**BERTscore > 0.75**	**Lev < 0.25**
MSRPARAPHRASE	81.8%	18.3%
QQP	64.1%	21.8%
PAWS	**100**%	**65.6**%
MSCOCO	**13.6**%	0.6%
OPUSPARCUS	36.0%	0.5%

Table 3.3 – Rough distribution of the BERTscore and the Levenshtein distance (Lev) computed on the corpora paraphrases pairs.

From the BERTscore, we can deduct that the pairs of sentences taken from MSCOCO or OPUSPARCUS are mostly not close semantically speaking. For MSCOCO, this is logical knowing how we created this dataset. For OPUSPARCUS, it may come from the sensitivity of the BERTscore with very short sentences (majority in the corpus).

Concerning the Levenshtein distance, these statistics show that the sentence pairs from the PAWS corpus are very close on the surface. In the same way, it is very logical knowing that PAWS has been created by word swapping techniques and contains sentences containing the same words.

Before digging into the parphrase generation task, we conducted a small experiment on paraphrase identification.

3.3.2 A Paraphrase Identification Experiment

The paraphrase identification task is a **classification task**. The goal is simple: from a pair of sentences as input, a binary classifier labels it as a paraphrase pair or not. As a first experiment, we tried to model a paraphrase classifier.

Paraphrase identification data

For this experiment, we used two paraphrase identification datasets that are composed of sentences pairs. Some are pairs of paraphrases - labelled as *1* - others are not - labelled as *0*.

It is important to notice that these corpora contain only valid sentences in terms of syntax and fluency. As a consequence, a good paraphrase identifier is a model that tells if two **correct** sentences are **semantically close** or not.

Text generation metrics

There exist several metrics to evaluate text generation tasks by comparing generated text with a gold target. It was the case for the RDF-to-text task presented in Chapter 2 and it is the case for machine translation and paraphrase generation too.

The goals of this experiment is to tell if the metrics used to evaluate the performances of paraphrase generators are good paraphrase identifiers.

Here we will evaluate the metrics BLEU and the BERT score as paraphrases identifiers.

Training datasets

To carry this experiment, we used two datasets: QQP and PAWS.

As a reminder, QQP is a corpus of questions extracted from Quora and PAWS is a corpus that contains adversarial paraphrases with a high lexical coverage. PAWS is by definition difficult for surface metrics like BLEU. The pairs of this datasets are hence challenging for models that do not capture non-local contextual information.

We give examples drawn from those two datasets in Table 3.1. These corpora are widely used paraphrase identification corpora but they can bu used as paraphrase generation corpora as well. We give more details on these datasets in Section 3.2.

Evaluation metrics

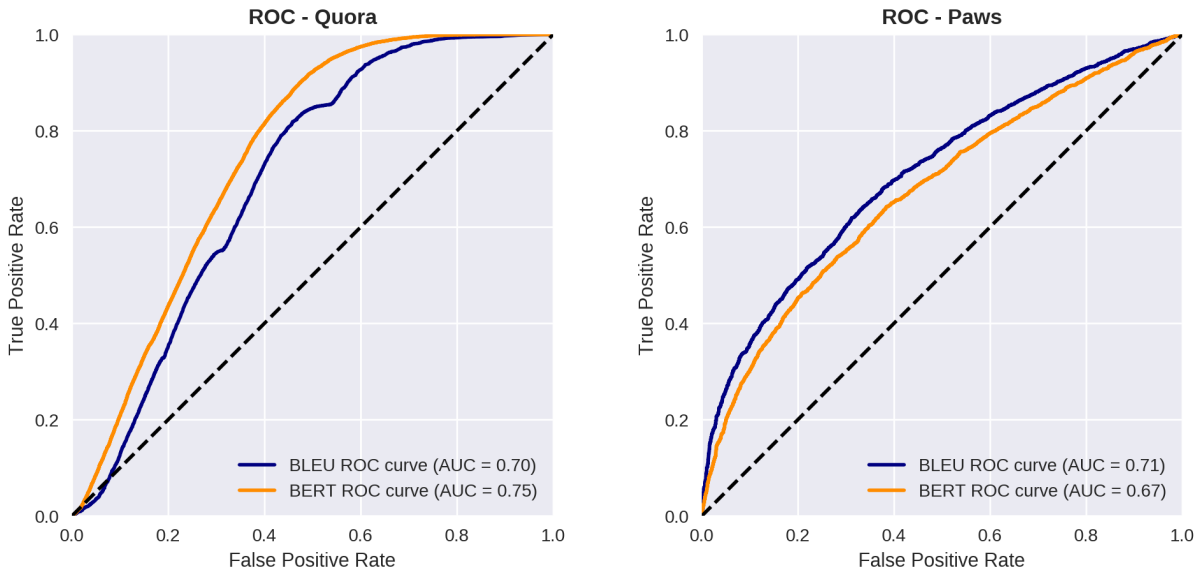
We evaluate the performances of the BLEU and the BERT scores as paraphrase identifier.

One can make a classifier out of a metric by setting a threshold. Sentences pairs associated with a higher score would be classified as paraphrases pairs. At the opposite, sentence pairs associated with a lower score would be classified as non-paraphrase pairs.

We can check the performances of the two metrics as classifier models by computing the Receiver Operating Characteristics (ROC) curve and the Area Under The Curve (AUC). A perfect classifier would have an AUC of 1 while a random classifier would only obtain 0.5.

Experiment results

We plot the ROC curve in Figure 3.2. For each corpus, we plot the ROC obtained by each model, corresponding to BLEU, the BERT score and a random classifier. We also report the AUC for each curves.



(a) QQPclassifiers ROC curve

(b) PAWSclassifiers ROC curve

Figure 3.2 – Paraphrase identification experiment with automatic metrics BLEU and BERT score on paraphrase identification corpora QQP and PAWS. The figure show the Area under ROC curve (AUC) on QQP (Subfigure (a)) and PAWS (Subfigure (b)). The metrics BLEU (blue curve) and BERT score (orange curve) are used as paraphrase identifiers. Both metrics are reasonable as paraphrase identifiers. On PAWS we observe a slight decrease in AUC that suggests that the metrics may fail to distinguish the harder adversarial examples.

This experiment shows that we have metrics that are reasonable as paraphrase identifiers The BLEU metric classifiers achieve AUC of 0.7 and 0.71 on QQP and PAWS respectively. The BERT score has an even higher AUC on QQP (0.75). However on PAWS, we observe a slight decrease (0.65). Zhang* et al. [2020] presented a similar experiment to evaluate the robustness of the BERT score and showed a performance drop of all the metrics on PAWS by suggesting that the metrics fail to distinguish the harder adversarial examples. Sentence pairs in PAWS are generated through word swapping that can lead to difficulties for a too simple classifier.

Conclusions

To conclude on this, we focused here on the paraphrase identification task. It is still an active topic of research and convey its own difficulties linked to the data and task complexity.

The task of paraphrase generation is however more difficult than the identification task. Indeed, indentifying the pairs of sentences that are paraphrase or not is just a binary classification problem. Also, it is often assumed that the pairs of sentences that we try to classify are valid sentences. There is no guarantee that a classifier trained on valid sentences would be efficient if deployed on ill-formed sentences. On the opposite, the paraphrase generation task implies, first to generate a valid sentence and second that this sentence is a paraphrase of another sentence. In the following subsection 3.4 we survey the existing paraphrase generation approaches.

3.4 Paraphrase Generators Overview

3.4.1 Rule-based and Statistical Approaches

Following the path of machine translation, the paraphrase generation literature first evolved from laboriously handcrafted linguistic rules [Carroll et al., 1999; Chandrasekar and Srinivas, 1997; McKeown, 1979; Meteer and Shaked, 1988] to more automatic and data-driven rules extraction methods [Callison-Burch, Koehn, et al., 2006a; Madnani and Dorr, 2010]. Like in machine translation, phrase-level substitution rules can be extracted by sub-sentence alignment algorithms from parallel corpora [P. E. Brown et al., 1993]. One can use different sources of data, the ideal being a human-labeled monolingual parallel corpus. *i.e.* a set of aligned sentences each being a proper variant of the others. Building such a dedicated corpus being a long and costly task, one usually transforms other corpora through a "pivot representation" [Barzilay and McKeown, 2001; Callison-Burch, Koehn, et al., 2006a; Chen et al., 2015; Ganitkevitch et al., 2013]. Alignment algorithms being quite robust, one can use even more lightly-related texts like news from the same period [Dolan and Brockett, 2005]. The strength of this approach is that large tables of phrase-level rewriting rules are relatively easy to extract even from lightly related texts.

The weakness of these approaches is that phrase-level rewriting rules alone are not able to build coherent sentences. A typical data-driven paraphrase generator used to be a mixture of potentially noisy handcrafted and data-driven rewriting rules coupled with a score that had to be optimized on real time though dynamic programming. But dynamic programming methods like *Viterbi* are constrained by the requirement of a score that decomposes into a sum of word-level or phrase-level criteria [Xu et al., 2016]. Some attempts were made to relax this constraint with search-based approaches [Chevelu et al., 2009; Daumé et al., 2009], but the global optimized criteria were simplistic and the obtained solutions were not suitable for a practical deployment.

3.4.2 Supervised Encoder-decoder Approaches

Just like machine translation, paraphrase generation benefited from deep neural networks and evolved to efficient end-to-end architectures that can both learn to align and translate [Bahdanau, Cho, et al.,

2016; Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al., 2017a]. Several papers like [Cao et al., 2017; Prakash, Sadid A. Hasan, et al., 2016a] set the paraphrase generation task as a supervised sequence-to-sequence problem. As confirmed by our experiments in Section 4.2, this approach is efficient for specific types of paraphrases, it is also able to produce relatively long-range transformations. But it also requires huge and high-quality sentence-level aligned datasets for training.

Chapter 4

Supervised Paraphrase Generators

Similarly to machine translation or RDF-to-text generation, paraphrase generation can be seen as a sequence to sequence problem.

Artificial neural networks achieved high performances on the machine translation task leading to a new approach to machine translation: *neural machine translation* (NMT). Indeed, the original encoder-decoder architecutre or seq2seq [Sutskever et al., 2014] enabled the power of deep neural networks to map sequences to sequences and outperform the phrase-based SMT systems. NMT has been an active topic of research since and the encoder-decoder architectures became huge.

A major drawback of using the advances of NMT directly to the task of paraphrase generation is that there is not as many aligned data for paraphrase generation as for machine translation. The corpora are smaller and/or more biased. The recent release of the QQP corpus and the existence of the MSCOCO dataset that can be used as an aligned paraphrase corpus, enabled the use of supervised encoder-decoder models for paraphrase generation.

In Section 4.1, we give an overview of the supervised encoder-decoder models used for paraphrase generation. We also highlight in Subsection 4.1.2 the reproducibility issues we encountered with the litterature models.

4.1 Overview of Supervised Paraphrase Generation

4.1.1 Survey

Like machine translation, paraphrase generation benefited from deep neural networks and evolved to efficient end-to-end architectures that can both learn to align and translate [Bahdanau, Cho, et al., 2016; Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez, et al., 2017b]. Several papers like [Cao et al., 2017; Prakash, Sadid A. Hasan, et al., 2016a] set the paraphrase generation task as a supervised sequence-to-sequence problem. As confirmed by our experiments in Section 4.2, this approach is efficient for specific types of paraphrases. It is also able to produce relatively long-range transformations and complex syntax structure, but it requires huge and high-quality sentence-level aligned datasets for training.

The paraphrase generation literature mostly reports their results on the MSCOCO and QQP datasets which are built respectively from image captions and duplicate questions. These datasets are very specific: MSCOCO is strongly biased toward image description sentences, and QQP is dedicated to questions.

To illustrate the datasets biases, a Transformer model trained on QQP typically transforms any input

into a question. For instance, from *"He is speaking on june 14."* it generates *"Who is speaking on june 14 ?"*.

We conducted experiments to reproduce the results of the supervised encoder-decoder models as reported in the literature. There is however no uniform experiment setup to directly compare the models and the experiments shown in the diverse papers. This leads to reproducibility issues that we detail in Subsection 4.1.2. On top of trying to reproduce the existing results, we conducted the experiments using a uniform framework in order to actually compare the results and extend them to all datasets. We present the results of those experiments in Section 4.2.

4.1.2 State of the Art Results Reproduction Issues

In the paraphrase generation literature, most of the papers report on MSCOCO and QQP. In table 4.1, we provide the BLEU scores as reported in [Egonmwan and Chali, 2019a; Fu et al., 2020; Gupta et al., 2018; Prakash, Sadid A. Hasan, et al., 2016a]. However, even if the dataset names coincide, and even if each evaluation methodology is correct on its own, the discrepancies between methodologies render these values impossible to compare with each other.

The strange gap between the residual LSTM performance of [Prakash, Sadid A. Hasan, et al., 2016a] and the one reported in [Fu et al., 2020] that tried to reproduce the results of the former can be explained by the fact that the first one is using the 2014 version of MSCOCO while the other is using the 2017 version. But we also found several other divergences:

— test sets splits
— sentence length shrinking
— vocabulary size
— tokenization strategies
— case

Regarding the sentence lengths, Prakash, Sadid A Hasan, et al. [2016b] and Gupta et al. [2018] shrunk all sentences to 15 words. Fu et al. [2020] set the maximum length to 16 while Egonmwan and Chali [2019b] set it to 15 and 10 respectively for the input and target sentences. Knowing that roughly 56% (resp. 5%) of MSCOCO target sentences are strictly longer than 10 (resp. 15) words, these small changes can have a great impact on the results.

The vocabulary considered also differs. Fu et al. [2020] used a vocabulary of $8k$ and $11k$ tokens from the train sets of QQP and MSCOCO respectively, whereas Egonmwan and Chali [2019b] had a vocabulary of approximately $15k$ words that was constructed on both the train and test sets.

The scripts used to compute metrics did also differ from one paper to another, and it is known that BLEU scores can vary widly with different parameterizations [Post, 2018].

For our reproduction experiments results, we used the code associated with the respective papers when available and otherwise, we tried as much as possible to reproduced the models of the literature faithfully. This allowed us to have the exact same preprocessing, training and testing pipeline for all our experiments. As a side effect, we provide a grounded benchmark between the methods that we could test.

Model applied on MSCOCO	*BLEU* ↑
RESIDUAL LSTM [Fu et al., 2020]	23.7
LBOW-TOPK [Fu et al., 2020]	25.3
RESIDUAL LSTM [Prakash, Sadid A. Hasan, et al., 2016a]	37.0
VAE-SVG-EQ [Gupta et al., 2017]	39.6
TRANSFORMER [Egonmwan and Chali, 2019a]	41.8
TRANSSEQ [Egonmwan and Chali, 2019a]	44.5

Model applied on QQP	*BLEU* ↑
RESIDUAL LSTM [Fu et al., 2020]	24.9
CGMH [Miao et al., 2018] - *weakly-supervised*	18.8
LBOW-TOPK [Fu et al., 2020]	26.2
VAE-SVG-EQ [Gupta et al., 2017]	37.1
TRANSFORMER [Egonmwan and Chali, 2019a]	39.0
TRANSSEQ [Egonmwan and Chali, 2019a]	39.8

Table 4.1 – Inconsistent BLEU scores as reported in several articles on paraphrase generation.

4.2 Paraphrase Generation Experiment

As supervised baselines, we trained three neural network architectures that were previsouly reported to achieve good results on MSCOCO and QQP, in particular, the Seq2Seq architecture, a *Residual* LSTM architecture [Prakash, Sadid A Hasan, et al., 2016b] and a TRANSFORMER model [Egonmwan and Chali, 2019a]. We extended the experiments to the other aligned corpora: MSRPARAPHRASE, OPUSPARCUS and PAWS.

4.2.1 Settings

To be more precise, we trained a 4-layers LSTM Seq2Seq with a bidirectional encoder and decoder using attention. This architecture is reported as SEQ2SEQ in the results. We trained a 4-layer Residual LSTM Seq2Seq as introduced by Prakash, Sadid A. Hasan, et al. [2016a] and reproduced by Fu et al. [2020]. This architecture is reported as RESIDUAL LSTM in the results. The results we obtained with this model are close to the ones reported by Fu et al. [2020].

Finally, we trained a TRANSFORMER using the *transformer_base* hyper-parameters set of Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N Gomez, et al. [2017b]. This architecture is reported as TRANSFORMER BASE in the results.

For all the encoder-decoder experiments, we used the *fairseq* framework [Ott et al., 2019] that implements the SEQ2SEQ and TRANSFORMER architectures. We added our own implementation of the RESIDUAL LSTM architecture.

For preprocessing, we used Moses tokenizer and subword segmentation following Sennrich et al. [2016] and using the *subword-nmt* library. The maximum sentence length is set to 1024 tokens which is the default setting in *fairseq*. For decoding, we did a beam search with a beam of size 5.

4.2.2 Results

We summarize the results of the experiments in Table 4.2. First, the three end-to-end architectures achieve very similar performances on all the corpora except on the MSRPARAPHRASE corporus.

On MSRPARAPHRASE, the TRANSFORMER model (BLEU: 20.7; METEOR: 21.6) is almost twice as good as the two other LSTM-based architecture BLEU: 11.6/10.5; METEOR: 12.6/11.2). The TRANSFORMER model achieves higher results because it handles better the long sentences of the MSRPARAPHRASE corpus. However, overall on MSRPARAPHRASE, the three encoder-decoder neural networks perform poorly. This result can be explained by the small number of training examples available on this corpus.

4.2.3 Analysis

We do not report significant differences between the neural networks architectures SEQ2SEQ, RESIDUAL LSTM and TRANSFORMER BASE.

Also the reported results on MSCOCO (BLEU $\approx$ 27) and QQP (BLEU $\approx$ 29) is coherent with the literature. Indeed, Fu et al. [2020] reports BLEU scores of 26.2 on QQP and 25.6 on MSCOCO for their

Corpus	**Model**	$BLEU \uparrow$	$TER \downarrow$	$METEOR \uparrow$	$BERT \uparrow$
MSCOCO	SEQ2SEQ	27.5	62.3	24.3	0.76
	RESIDUAL LSTM	26.9	63.3	24.2	0.76
	TRANSFORMER BASE	26.9	63.3	24.2	0.76
QQP	SEQ2SEQ	29.2	60.3	30.7	0.8
	RESIDUAL LSTM	28.4	59.1	30.2	0.8
	TRANSFORMER BASE	29.1	59.5	30.5	0.8
OPUSPARCUS	SEQ2SEQ	8.4	79.3	13.8	0.69
	RESIDUAL LSTM	8.1	78.6	14.3	0.7
	TRANSFORMER BASE	8.1	84.3	13.9	0.7
PAWS	SEQ2SEQ	44.0	36.8	39.2	0.92
	RESIDUAL LSTM	43.6	37.1	38.9	0.92
	TRANSFORMER BASE	42.4	37.6	38.9	0.92
MSRPARAPHRASE	SEQ2SEQ	11.6	89.5	12.6	0.53
	RESIDUAL LSTM	10.5	93.7	11.2	0.52
	TRANSFORMER BASE	20.7	76.7	21.6	0.65

Table 4.2 – **Experiments summary.** Symbol '↑' means that higher value is better. The results are in same range for the tree models for all corpora except for MSRPARAPHRASE. On MSRPARAPHRASE the TRANSFORMER model outperforms the other two. This can be explained by the best performances of the TRANSFORMER on long sentences.

model. However Egonmwan and Chali [2019a] reports much higher results (BLEU = 41.8 on MSCOCO and BLEU = 39.0 on QQP). We explain this gap by the different experiment settings that render the expriments incomparable.

Chapter 5

Search-based Approaches

In Chapter 4, we trained sequence-to-sequence neural networks to generate paraphrases. This method is very efficient for many text generation tasks. The main drawback of this strategy is its requirement to have a large aligned dataset. We also noticed that even if a model trained on a specific dataset performs well on that dataset, it remains very biased. The available aligned paraphrase corpora presented in section 3.2 are often biased toward specific problems like *question answering* or *image captioning*.

With the lack of generic aligned datasets, it remains challenging to train generic paraphrase models in a supervised manner. For this reason, we studied a different approach: a search-based approach.

Search-based approaches provide a better control of the output. They are often used to integrate constraints in text genetation like in CGMH [Miao et al., 2018].

In this chapter, we detail the study of a search-based generation scheme where candidate paraphrases are generated by iterated transformations from the original sentence. The motivation behind this approach is to have a more generic model and to be independent of the requirement of a huge aligned training dataset.

The chapter is divided in several sections. In Section 5.1, we introduce the casting of the paraphrase generation task as a tree-search problem. In Sections 5.2 and 5.3, we detail our search-based algorithms MCPG and PTS. In Section 5.4 we provide an extensive set of experiments on the five paraphrase datasets. And finally we conclude in Section 6.3

5.1 A Paraphrase Tree Generator

We model paraphrase generation as a sequence of editions and transformations from a source sentence into its paraphrase. We only consider local transformations, *i.e* replacement of certain words or group of words by others that have the same or similar meanings.

The concept is easy. Starting from a source sentence, that we are willing to paraphrase, we create a generator of paraphrase candidates using local transformations. This generator can be seen as generating a tree of candidates, where each node is a possible paraphrase of the source sentence and each edge is a transformation made to go from the source node to the target node. The root of the tree being the source sentence, the deeper we go in the tree, the more different the candidate paraphrase is from the source sentence. In a second stage, we are looking for the best paraphrase candidate or the best node in the tree using a search algorithm.

This paraphrase tree generator leverages a database of phrasal and lexical transformation called PPDB [Pavlick et al., 2015] presented in subsection 5.1.1. In the following subsection 5.1.2, we will

show some statistics about the generated paraphrase spaces according to each paraphrases corpora. The paraphrase scorer is detailed in section 5.2.3

5.1.1 PPDB: The Paraphrase Database

The *Paraphrase Database* (PPDB) [Pavlick et al., 2015] is a large collection of scored paraphrase rules that was automatically constructed from various bilingual corpora using a pivot alignment method [Callison-Burch, 2008]. The database is divided into increasingly large and decreasingly accurate subsets labelled S, M, L, XL and XXL. We used the XL subset, and we removed the rules labeled as *"Independent"*. This left us with a set of 5.5 million rewriting rules. We give some examples of these rules in Figure 5.1. PPDB provides also several informations on the transformation rule. In particular, it includes the *PPDB2.0Score* which is the score that was used to rank the rules in the database.

PPDB 2.0 score is a supervised scoring model introduced by Pavlick et al. [2015] when introducing the second version of PPDB, PPDB 2.0. Pavlick et al. [2015] collected 26455 humanly annotated paraphrase pairs from PPDB. They then fitted a regression of the human judgements using 209 features detailed in the paper. The features are lexical overlap features, features derived from WordNet, distributional similarity features or the cosine similarity of the word embeddings.

```
1 [JJ] ||| unavoidable ||| inevitable ||| PPDB2.0Score=4.39597
    ...
2 [VB] ||| ill-treatment ||| mistreatment |||
    PPDB2.0Score=4.69058
3 [JJ] ||| authorized ||| authorised ||| PPDB2.0Score=4.39584
    ...
4 [PP] ||| in different parts of the world ||| in various
    parts of the world ||| PPDB2.0Score=5.90031 ...
5 [NP/VP] ||| one of the key factors ||| one of the most
    critical factors ||| PPDB2.0Score=5.73433
6 [RB] ||| intentionally ||| deliberately |||
    PPDB2.0Score=4.59228
```

Figure 5.1 – **PPDB transformations sample.** Six lines sampled from the PPDB database. The transformation rules correspond to the second and third fields. PPDB also include several informations. We use the *PPDB2.0Score* in the fourth field.

By iteratively applying the rules from a source sentence like illustrated in figure 5.2, we obtain a vast lattice of candidate paraphrases. Some of these candidates like "*he's **talking** on june 14*" are well-formed, but many are syntactically broken, like "*he is **speak now** on june 14*".

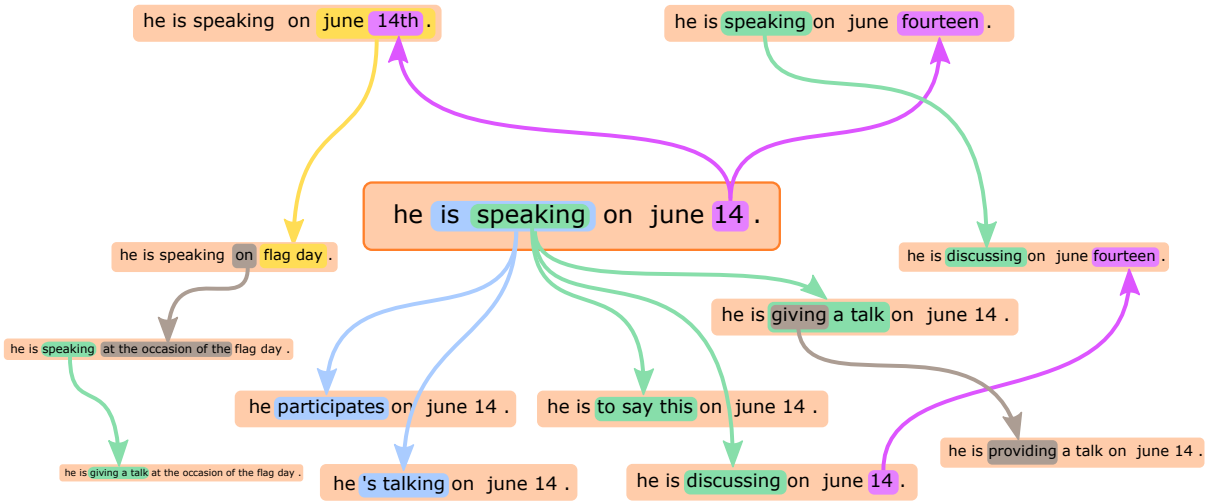


Figure 5.2 – **Example of a generated paraphrase space.** Starting from the source sentence (the root) *"he is speaking on june 14."*, the paraphrase generator applies PPDB transformations represented by the arrows in the figure. A transformation can be lexical or phrasal. Each transformations leads to a node in the generated tree representing a potential paraphrase of the source sentence. By applying one or many transformations to the source sentence, the model generates many paraphrase candidates.

5.1.2 Statistics on the Paraphrase Spaces

The number of rules that apply depends on the source sentence's size and the words it contains. For instance, on MSRPARAPHRASE dataset, sentences are quite long and the median number of PPDB-XL rules that apply is around 450. After two rewriting steps, the median number of candidates is around 10^5, and by iterative rewriting, we quickly reach a number of paraphrase candidates that is greater than 10^8

For a unique source sentence, the model can generate many candidates. The next phase was to develop and implement a good search algorithm to choose the best paraphrase among the lattice of candidates. In the following Section 5.2 and 5.3 we introduce two search- strategies: MCPG and PTS.

5.2 Monte-Carlo Paraphrase Generator

Following the idea of [Chevelu et al., 2009], we first experimented *Monte-Carlo Tree Search* (MCTS) to explore the PPDB lattice.

5.2.1 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) is a heuristic search algorithm that has three key ingredients. First, a bandit policy applied at each node of the search tree selects the most promising paths. Second,

randomized Monte-Carlo explorations (roll-outs) estimate the quality of the selected paths. Finally, a reward function is backpropagated along the paths to update the bandit.

The implementation of MCTS should also have a cache system to avoid duplicate evaluate of paths. Because, in our case the explored tree is rather a lattice and several branches can lead to the same paraphrase.

The historical version of MCTS was called UCT for *Upper-Confidence for Trees* [Kocsis and Szepesvári, 2006]. It was based on a deterministic optiministic bandits and designed mostly for problems like computer go where the evaluation of positions was binary and only possible on the leaves. Several improvements have been added afterwards:

— the RAVE heuristic which reinforce good positions already in the cache [Gelly and Silver, 2007]

— the bandit is replaced by an RL-trained policy network by Silver et al. [2016]

We opted here for a randomized bandit policy called EXP3 [P. Auer et al., 2002]. It samples sub-trees according to a Gibbs distribution.

$$P_i = \frac{exp(w_i/\tau)}{\sum_j exp(\lambda \cdot w_j/\tau)} \tag{5.1}$$

The weights w_i of each sub-tree i is first initialized to a bias value and then reinforced or lessened according to the feedback thus making the exploration focus towards the most promising options. The temperature parameter τ controls the balance between exploration and exploitation: when τ is small, the distribution is uniform and the algorithm explores a lot. Higher values tend to peak the distribution toward the most rewarding options. We have set this value to 0.1 in our experiments. This policy was combined with a RAVE heuristic to speed up the algorithm by also considering paths out of order when updating node.

Our model *Monte-Carlo Paraphrase Generator* uses MCTS for the task of paraphrase generation.

5.2.2 MCPG

The MCPG algorithm is described as pseudo-code in Algorithm 1 and illustrated in Figure 5.3.

The paraphrase generation task has a few specificities that require adaptations from the original MCTS algorithm. First, the average rewriting graphs are not very deep, but their branching factor vary and it can be very high on some instances thus rendering the exploration of all sub-trees intractable.

In MCTS only the leaves are evaluated while there is the possibility to score any node as each node is a paraphrase candidate although on some examples it may be misleading. For instance, when a transformation rule changes the singular to the plural form of a subject in the source sentence, another transformation rule is needed to change the verb form accordingly. We call it a *two-steps transform*. The candidate with the two transformations is valid but the intermediate candidate with only the first transformation is incorrect. Thus this first transformation can be scored as bad even if in theory it is completely fine as it leads to good candidates.

Also, to reduce complexity and to avoid a semantic drift, we forbid the modification of a word that was already rewritten.

Finally, from each node except the root, a *stop* action is playable to evaluate the paraphrase. Therefore, the best paraphrase can never be the source sentence.

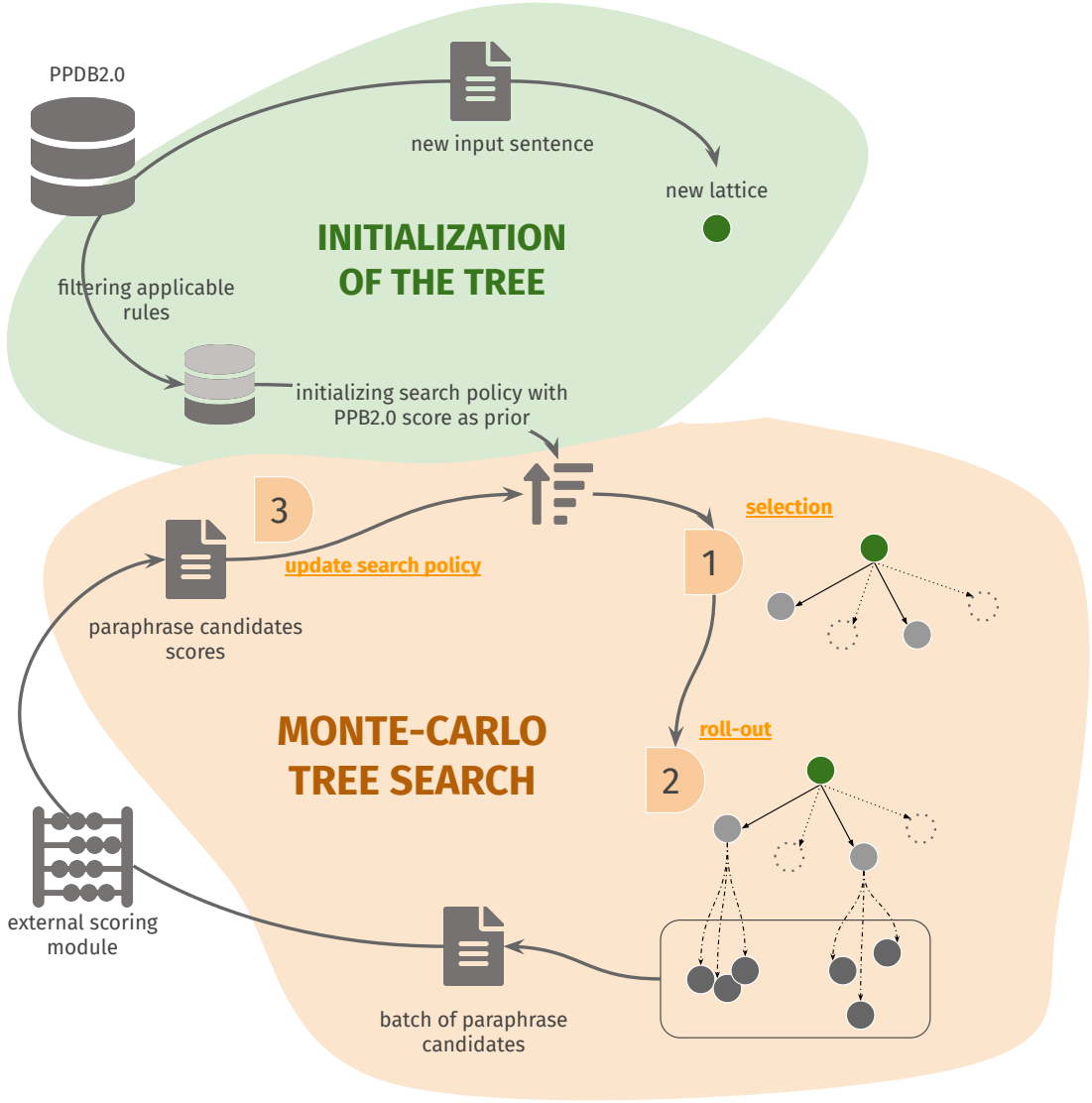


Figure 5.3 – **Overview of the MCPG program.** The first phase is depicted in green. MCPG first loads the source sentence to paraphrase and filters the applicable PPDB rules. It initializes the root of the search tree and the search policy with the *PPDB 2.0score* as prior. The second phase depicted in orange in the figure is the actual search. It is a loop that consists of 3 main steps. 1. Selection : MCPG applies the policy to choose nodes to explore. 2. Roll-out : from the selected nodes, MCPG applies the policy until leaves in the tree. The corresponding leaves are the paraphrase candidates that are scored by an external scoring module. 3. Update search policy : MCPG updates the policy with the computed paraphrase candidates scores and loops to the first step. This loop stops when the resources are consumed or when the whole tree has been explored.

Algorithm 1: MCPG

```
Input: input_sentence
candidates ← REWRITE(input_sentence)
depth ← 1
while time/space < budget do
    while time/space < layer-budget do
        nodes ← MAB.SELECTS_FROM(candidates)
        paths ← ROLLOUT_FROM(nodes)
        scored ← scored ∪ NN.SCORE(paths.leaves)
        Backward node rewards along paths
        MAB.UPDATES_POLICY(node rewards)
    layer_best ← ARGMAX(scored)
    candidates ← candidates ∪ REWRITE(layer_best)
    depth ← depth + 1
return ARGMAX(all scored nodes)
```

In figure 5.3, we detail all the steps of the program. MCPG was implemented in *C++* and is inspired from the source code of Chevelu et al. [2009].

At the loading of MCPG, the program loads the PPDB database in memory. Then, for each new input source sentence, the program filters the PPDB database keeping only the rules applicable to this particular sentence. Additionaly, the program initiates a new search policy for the new sentence leveraging the *PPDB2.0Score* associated with every applicable rule. This is what we call the *initialization of the tree* phase. It is represented in green in the overview illustation Figure 5.3.

Then – represented in orange in the illustration – the algorithm proceeds by epochs. The first epoch samples candidates without replacement according to the prior, and the next epochs takes the score feeback into account to focus the search on promising sub-trees. We have set the epoch to 10000 to balance between CPU and GPU. The prior bias is an important hyper-parameters. At each epoch, the program applies the steps of the MCTSalgorithm:

1. selection
2. roll-out
3. updating the policy

We developed a new C++ implementation of the algorithm described in Chevelu et al. [2009]. This version includes some code optimizations and caching techniques to speed up computation.

MCTS needs a reward function to update its policy at each epoch. We implemented an external scoring module as illustrated on the bottom left in Figure 5.3. More about this score will be given in subsection 5.2.3.

From a technical perspective, the MCPG program and the external scoring module communicate with text files directly in the file system. To launch an experiment we launch a wrapper script that launches in parallel the MCPG program and the scoring module. In Figure 5.4, we display the typical folder structure

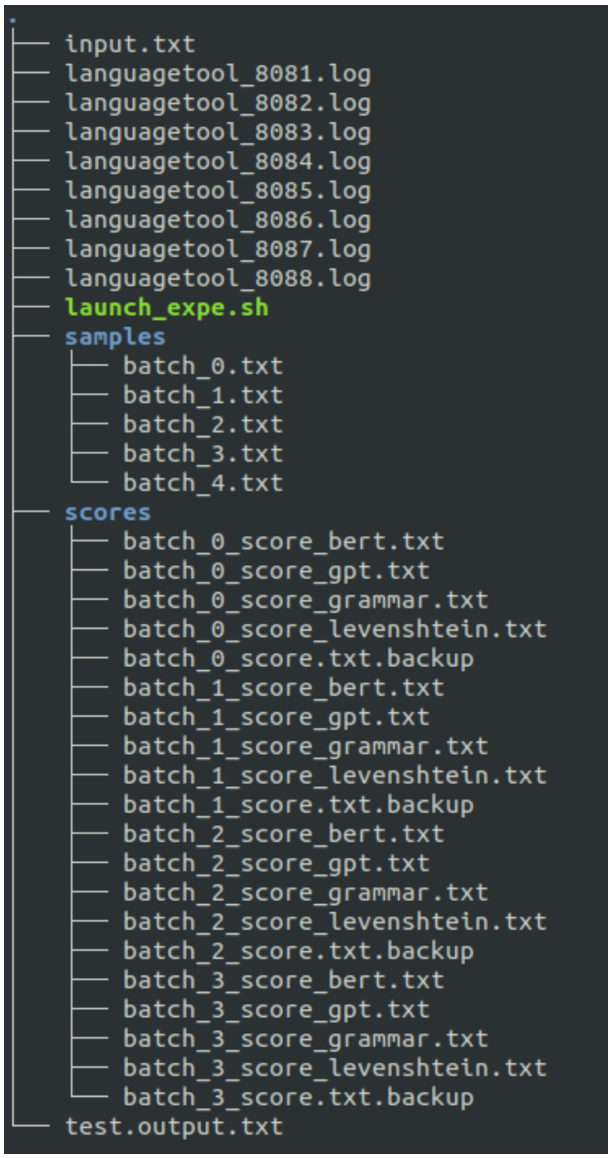


Figure 5.4 – Typical folder structure for an experiment with MCPG. The scores and samples folders allow the search program and the score function program to communicate the batches to be scored and the corresponding scores. The two programs that run in parallel can be launched from a *bash* script for example. Here it is the role of the *bash* script *launch_expe.sh*. The sentences to be paraphrased are also stored in a text file: here *input.txt*. And the generated paraphrases are stored in another text file: here *test.output.txt*. The program can also generate different log files.

for an experiment with MCPG.

To rank the paraphrase candidates, we developped a paraphrase score function that we try to optimize during the search. The vanilla MCTS algorithm is not designed for multi-objective problems. Hence, we needed to combine our criteria, namely the semantic similarity, the syntax correctness and the surface diversity into a single criterion. We detail the scoring function in the next subsection 5.2.3.

5.2.3 Scoring Function

In this subsection, we detail how we developed a paraphrase score. As presented in Chapter 3, the paraphrase generation task combines Natural Language Understanding (NLU) and Natural Language Generation (NLG). In particular, we saw in the experiment on paraphrase identification in subsection 3.3.2 that the metrics used to evaluate the paraphrase generation make reasonably good paraphrase classifiers.

On the basis of these remarks, we consider three axes for defining a good paraphrase.

Firstly, we need to maximize the semantic similarity between the source sentence and the generated paraphrase. Indeed, a paraphrase must have the same meaning as the source sentence, this is the NLU part of the score function.

Secondly, the generated paraphrase must be a valid sentence. This is the NLG part. We therefore need to add a syntax correction dimension to be maximized.

In terms of semantic, the best paraphrase of a sentence is the sentence itself. To make sure that the generated paraphrase has a different surface from the source sentence, we add a surface diversity dimension.

As it depends on the type of text we consider that may be spoken or written, casual or formal. It is not easy to define a universal semantic distance or a universal scale of well formed syntax. However, recent advances in NLP with neural networks like BERT and GPT-2 trained on huge corpora have led to the development of metrics that can act as good proxies for these ideal notions.

For each dimension, we detail in the following Sections 5.2.3, 5.2.3, 5.2.3 the scores we used to quantify each criterion. In Section 5.2.3, we introduce our scoring function that combines the three criteria.

Semantic

For the semantic distance, a quick experiment confirms that the BERTscore [Tianyi, 2019; T. Zhang et al., 2019] performs well on difficult paraphrase identification tasks. The BERTscore is an F1-measure over an aligment of the BERTcontextual word embeddings of each of the sentences. We opted for the BERTscore between the source sentence and paraphrase candidate (denoted BERT_s) as our semantic score.

We introduced the BERT score in subsection 2.1.4 and Section 3.3.

Syntax

Regarding the syntax quality, the perplexity of a language model is a good ranking criterion. The GPT-2 model [Radford et al., 2019] is a large pre-trained language model that leverage the TRANSFORMER model. It achieves state-of-the-art results on several NLP tasks.

Although, as illustrated in Table 5.1, in some cases, a rule-based spell-checker may detect errors that GPT-2 would miss (but the reverse is also true). We hence opted for GPT-2 as a primary criterion for syntax quality, combined with the LANGUAGE-TOOL spell-checker [Naber, 2003] that we only used on a second stage for performance reasons.

Diversity

The lexical distance is important to ensure the diversity of the produced paraphrases. It is however simple to handle. Some authors use the BLEU surface metric [Miao et al., 2018], we opted here for the normalized character-level Levenshtein edition distance.

Source	the agency could not say when the tape was made , though the voice says he is speaking on june 14 .				
Policy	**Candidate Sentence**	BERT$_s$	GPT-2	Lev_s	ERRS
High BERT → **conservative**	the agency could not say when the tape was made , though the voice says he is talking on june 14 .	**0.99**	4.23	0.04	0
High *Lev* → **risk prone**	the organizations and agencies could just 'm saying when- ever the tape-based was provided presentation there are , however the express opinions informed that he was found talking pertaining end-june fourteen .	0.60	7.83	**1.0**	0
High GPT-2 → **bad syntax**	the agencies was not able say when the tape been given currently undertaking though the voice indicated he talking on june 14 .	0.82	**7.10**	0.55	2
Spell & Grammar errors	the organisation are incapable of 're saying when the tape is set out , despite the fact that the voice just said he have a conversation on june 14 .	0.73	5.49	0.79	**2**
Balanced performances	the organization could not say when the tape was made , although the voice indicates that he is talking on june 14 .	0.95	4.34	0.28	0
MCPG**output**	the organization could not say when the tape was made , though the voice indicates that he is talking on june 14 .	0.96	4.36	0.26	0
Aligned target	the agency could put no exact date on the tape , though the voice says he is speaking on june 14 .	0.87	4.77	0.22	0

Table 5.1 – An example of a source sentence sampled from the MSRPARAPHRASE train set with some candidate paraphrases generated by our system. The transformation made on the original sentence are highlighted in blue. The spell and grammar errors detected by the LanguageTool spell-checker are highlighted in red. We display here a candidate that maximizes the BERTScore and is, as a consequence, very conservative; a candidate that maximizes the Levenshtein distance and takes a lot of risks; a candidates with a bad syntax; a candidates that illustrates the utility of the spell-checker; our equilibrium goal; the paraphrase generated by our model MCPG and the target in the dataset.

Source	it 's going to be all right .				
Policy	**Candidate Sentence**	BERT$_s$	GPT-2	Lev_s	ERRS
High BERT → **conservative**	it 's gonna to be all right .	**0.96**	5.20	0.07	0
High *Lev* → **risk prone**	this was proceeding to be considered is everything access rights .	0.38	7.22	**1.0**	0
High GPT-2 → **bad syntax**	it was 're gonna be principle underlying correct .	0.52	**9.08**	1.0	0
Spell & Grammar errors	it it is going to be all right .	0.89	3.95	0.14	**1**
Balanced performances	it 's gonna be all right .	0.96	4.74	0.17	0
MCPG**output**	it 's gonna 's going to be all right .	0.94	4.57	0.31	0
Aligned target	listen , it 's gonna be okay .	0.78	5.27	0.79	0

Table 5.2 – An example of a source sentence sampled from the OPUSPARCUS train set with some candidate paraphrases generated by our system.

The three criteria equilibrium

The balance between these criteria is difficult to obtain.

In order to illustrate the choice of our score function, we dispay, in Tables 5.1 and 5.2, several examples of candidate paraphrases from two batches generated by the MCPG module for sentences drawn respectively from MSRPARAPHRASE and OPUSPARCUS . For each sentence, we detail the different parts of our score, namely the BERT_s score, the perplexity score from the GPT-2 model, the Levenshtein distance Lev_s and the number of new spell and grammar errors detected by the LANGUAGE-TOOL system in the candidate sentence.

Table 5.1 illustrates their impact on a sentence taken from the train set of MSRPARAPHRASE. The candidate examples in Tables 5.1 and 5.2 underline the tough dilemma between maximizing the semantic similarity (safe and conservative policy) and maximizing the lexical distance (risk-prone). The third and fourth examples underline the utility of the spell-checker:some low-perplexity examples are ill-formed.

The easiest option was a linear combination but after some quantitative analysis of the scores distributions, we realized that it was easy to maximize the score by just applying a lot of editions to the source sentence.

We hence opted for the following polynomial:

$$\alpha \cdot \text{BERT}_\text{S} + \beta \cdot \text{Lev}_\text{S} \cdot \text{BERT}_\text{S} - \gamma \cdot \text{GPT2} \tag{5.2}$$

where the product $\text{Lev}_\text{S} \cdot \text{BERT}_\text{S}$ is intended to avoid a trivial maximization of the score by applying a lot of editions to the source sentence. After a few experiments on train sets, we tuned empirically the weights to $\alpha = 3, \beta = 0.5$ and $\gamma = 0.025$ in order to obtain a balance as the one described on Tables 5.1 and 5.2.

In both tables, we display the paraphrase generated by MCPG using the scoring function defined previously. For information purposes, we also display the gold target reference associated with the two sentences in their respective datasets. Our scoring function achieves a good balance between the criteria. However, as illustrated by the example sampled from OPUSPARCUS in Table 5.2, the MCPG output is sometimes not perfect.

In fact, MCPG is not really suited for such a multi-objective problem like paraphrase generation. In the next Section 5.3, we introduce a second model to correct the MCPG defaults.

5.3 Pareto Tree Search

We have previously highlighted that MCPG, the method based on the Monte-Carlo Tree Search algorithm, does not suit well the problem of paraphrase generation.

First in the case of the paraphrase generation problem and in particular in the search-based framework we have chosen where for each source sentence we construct a lattice of candidate paraphrases (the nodes) associated with local transformations (the edges), each node is also a leaf. In practice, MCTS is an efficient algorithm on combinatorial problems where evaluation is only possible on the leaves of the tree. In our particular case where all nodes are leaves, we add an extra complexity to the search by applying many transformations to the source node to evaluate it, whereas we could directly evaluate it without doing

a roll-out because it is already a leaf. As pointed out in Section 5.2.2, some transformations are two-steps transformations, meaning that the candidate after the two transformations is syntaxicly valid while the candidate with only the intermediate transformation is syntaxicaly broken. In practice two-steps rewriting is rare and the branching factore is too high to explore all the two-steps rewritings.

Also, MCPG can only optimize a single score. Since we want to evaluate the candidate paraphrases along three axes that we defined in section 5.2.3 we had to scalarize the different subscores and set the weights in order to define a score function. This work is tedious and inexhaustive. Hyperparameters are difficult to optimize automatically.

As detailled in Section 5.2.3, the ideal would be to have a score function that sufficiently models the balance between all criteria. But in practice, this balance is unstable because it is much easier to optimize the surface diversity than the semantic similarity, especially if we do roll-outs. The examples Tables 5.1 and 5.2 illustrate this problem well.

We have developed a new search algorithm presented in this section to tackle both problems.

5.3.1 From a Single Objective to the Pareto Front

When developing the score function presented in equation 5.2, we plotted some batches of candidates as scatter plots. Each candidate is a point in a three-dimension space, one dimension by subscore. After that, we chose a type of score function suited to model the paraphrase score.

In fact, we can explore this idea even further. We can directly compute the Pareto front of the batch of candidate paraphrases.

Let's define the concept of *Pareto efficiency* and the *Pareto front*.

Pareto efficiency

As defined in [Osborne and Rubinstein, 1994]
Let N be a finite set and let $X \subseteq \mathbb{R}^N$ be a set.
Then $x \in X$ is **Pareto efficient** if there is no $y \in X$ for which $y_i > x_i$ for all $i \in N$;
$x \in X$ is **strongly Pareto efficient** if there is no $y \in X$ for which $y_i \geq x_i$ for all $i \in N$ and $y_i > x_i$ for some $i \in N$.

In our case, we have a set of paraphrases that are scored with three subscores. Our batch of candidates is a finite set $X \subseteq \mathbb{R}^3$. A paraphrase A is *Pareto efficient* is there is no other paraphrase B such that all the subscores of B are higher than the subscores of A. All the paraphrases of the batch that are efficients constitute what we call the *Pareto front*. The other paraphrases are said to be *dominated*.

Pareto front

The **Pareto front** (also frontier or set) is the set of all Pareto efficient samples.

In practice, by plotting the distributions of the scores like in Figure 5.5, we noticed that most of the candidates were dominated in the Pareto sense: it was possible to eliminate most of the candidates paraphrases without any hyperparameter tuning.

Hence, we adapted MCPG to explore the paraphrase lattice and recover an approximation of the Pareto front, postponing the balance between the criteria as a quick post-optimization stage. This is the *Pareto Tree Search* (PTS) algorithm. We detail PTS in the next Subsection 5.3.2.

5.3.2 PTS

The PTS algorithm is described as pseudo-code in algorithm 2 and illustrated in Figure 5.6.

Algorithm 2: PTS

```
   Input: input_sentence
 1 candidates ← REWRITE(input_sentence)
 2 depth ← 1
 3 while time/space < budget do
 4 |   while time/space < layer-budget do
 5 |   |   batch ← SAMPLE(candidates)
 6 |   |   scored ← scored ∪ NN.SCORES(batch)
 7 |   layer_front_set ← PARETO-FRONT(scored)
 8 |   candidates ← REWRITE(layer_front_set)
 9 |   depth ← depth + 1
10 return PARETO-FRONT(all scored nodes)
```

For PTS, the initialization phase is identical to that of MCPG.

The search process is iterative over the depth of the tree. In other words, at each iteration we go down one step in the tree.

The idea is simple, we consider all the paraphrases or a fixed size sample if the sentence is very long (or if there are many applicable rules) by applying all the possible transformations (or a sample). This is represented as *step 1* in Figure 5.6. These paraphrases constitute the current batch of candidates. This batch is scored with the same tools as for MCPG. This is *step 2*.

Using the score vectors associated with each candidate, we filter out the Pareto front and the dominated paraphrases. This is *step 3* in the illustration. The dominated paraphrases are removed and these branches will not be explored further.

In the next interations, we start again from the current Pareto front. We apply the applicable transformations on the already scored paraphrases (the already explored nodes), we score and we filter the dominated paraphrases in the Pareto sense.

This orange loop in Figure 5.6 corresponds to the two *while* loops on line 16 and 17 of Algorithm 2. In practice, the two steps of creating new candidates by applying rules and scoring them is done iteratively in smaller batches because of the space limit of the scoring module. That is why we have two loops in the algorithm. For the convenience of the reader, only one loop is shown in the illustration.

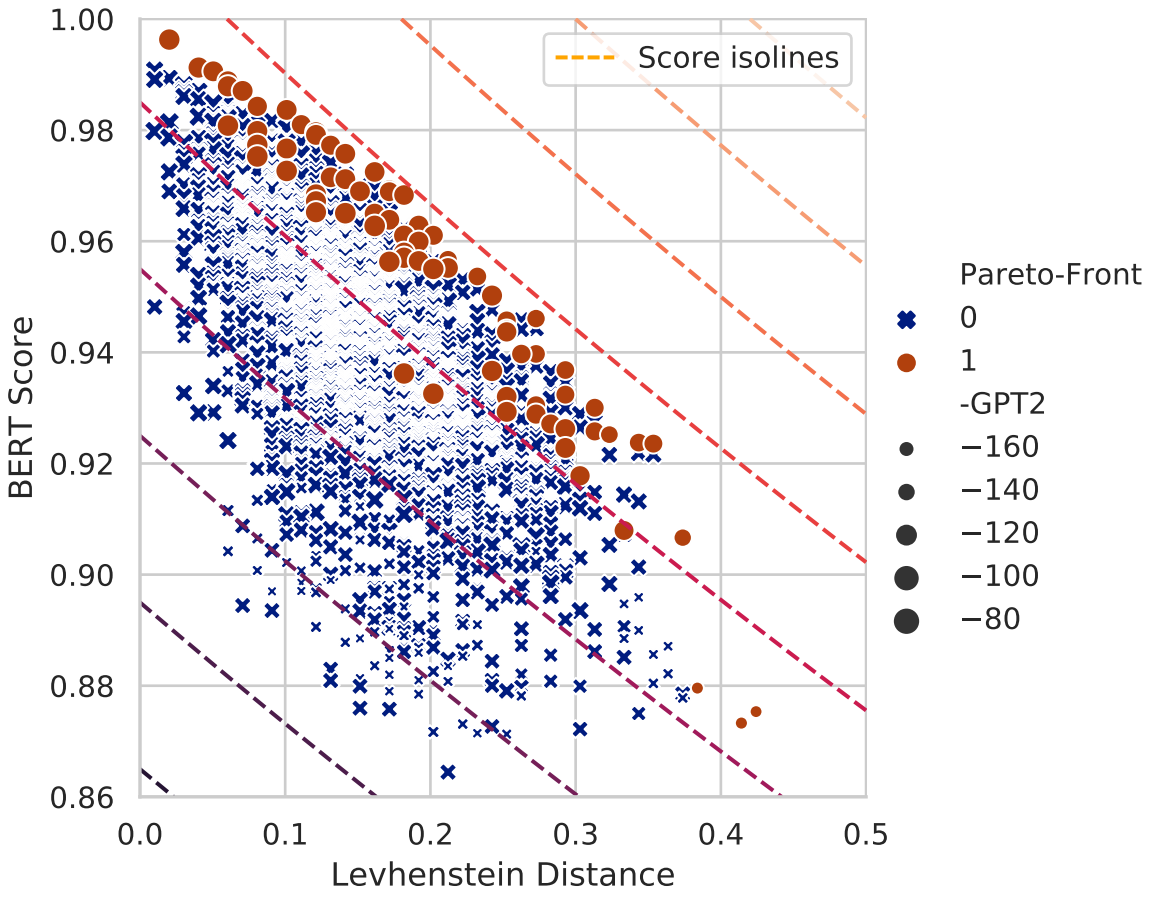


Figure 5.5 – **Candidates scores distribution sample and Pareto front.** The figure displays an example of the subscores distributions for a batch of paraphrases candidates. The candidates that belong to the Pareto Front are depicted in orange circles. One can notice that the majority of the candidates are Pareto dominated.

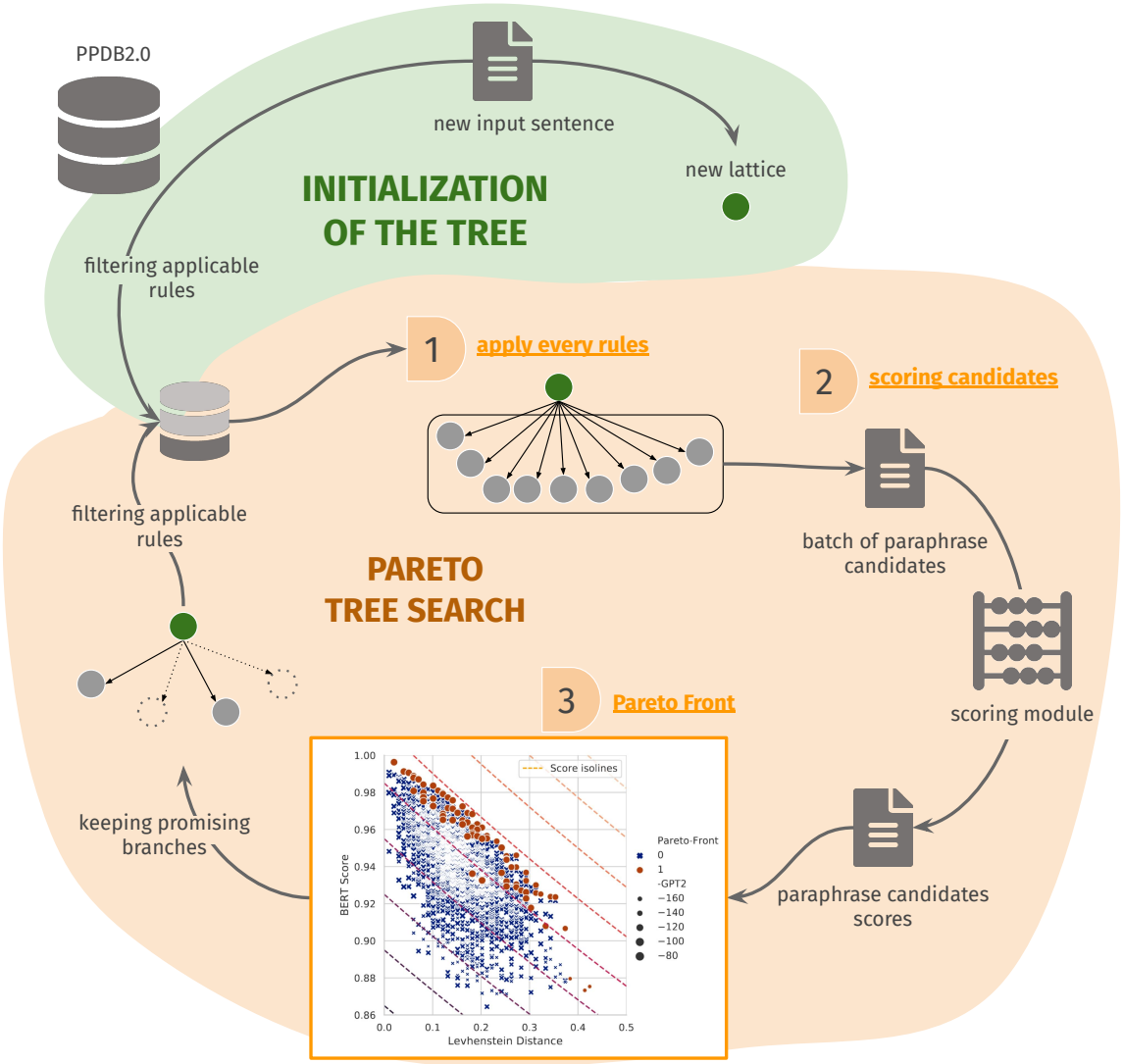


Figure 5.6 – Overview of the PTS program.

When the search is complete, we have extracted a set of candidate paraphrases that form an estimate of the Pareto front. A final step is therefore necessary to select the best paraphrase to be generated by the model. This is the selection phase. In fact, different selection policies can be applied. And we will see later that we can easily use native selection policies as a baseline to evaluate the model. For our experiments, we optimized the score developed for MCPG as a selection criterion within the Pareto front.

5.3.3 $\mathcal{E}$-PTS

PTS is not guaranteed to deliver the global Pareto Front due to the two-steps rewriting rules ans the sampling procedures. In order to deal with two-steps rewriting we also developed a relaxed version of PTS: $\mathcal{E}$-PTS inspired by the work on the Pareto front approximation presented by Legriel et al. [2010].

$\mathcal{E}$-Pareto efficiency

Let N be a finite set and let $X \subseteq \mathbb{R}^N$ be a set.
Then $x \in X$ is **$\mathcal{E}$-Pareto efficient** if there is no $y \in X \setminus \{x\}$ for which $y_i > x_i + \epsilon$ for all $i \in N$;
A point x that is not $\mathcal{E}$-Pareto efficient is said to be **$\mathcal{E}$-Pareto dominated**. $x \in X$ is **$\mathcal{E}$-Pareto dominated** in X if $\exists y \in X \setminus \{x\}$ for which $x_i + \epsilon < y_i, \forall i \in N$

When $\epsilon = 0$, $\mathcal{E}$-Pareto and Pareto are equivalent. The higher ϵ, the more points are included in the $\mathcal{E}$-Pareto front and the less points are dominated.

$\mathcal{E}$-net

Let N be a finite set and $X, X' \subseteq \mathbb{R}^N$ be sets. X is an **$\mathcal{E}$-net** of X' if $\forall x' \in X' \exists x \in X$ for which $|x_i - x'_i| \leq \epsilon \ \forall i \in N$

Lemma 1

Let N be a finite set and $X, X' \subseteq \mathbb{R}^N$ be sets.
If $x \in X$ is $\mathcal{E}$-Pareto dominated in X then $\forall x' \in X$ such that $|x_i - x'_i| \leq \epsilon \ \forall i \in N$ x' is Pareto-dominated in X.

Proof of Lemma 1

Let N be a finite set and $X, X' \subseteq \mathbb{R}^N$ be sets.
Let $x \in X$ be $\mathcal{E}$-Pareto dominated in X: $\exists y$ such that $\forall i \in N y_i > x_i + \epsilon$.
Let $x' \in X$ such that $|x'_i - x_i| \leq \epsilon \ \forall i \in N$.

$\forall i \in N, |x'_i - x_i| < \epsilon$
$\Rightarrow \forall i \in N, x'_i - x_i < \epsilon$
$\Rightarrow \forall i \in N, x'_i < x_i + \epsilon < y_i$
$\Rightarrow x'$ is dominated by y.

From this concept of relaxed Pareto front, we relaxed the PTS algorithm into the $\mathcal{E}$-PTS algorithm. The latter is presented in pseudo-code in Algorithm 3.

Algorithm 3: $\mathcal{E}$-PTS

```
Input: input_sentence
candidates ← REWRITE(input_sentence)
depth ← 1
while time/space < budget do
    while time/space < layer-budget do
        batch ← SAMPLE(candidates)
        scored ← scored ∪ NN.SCORES(batch)
    layer_front_set ← E-PARETO-FRONT(scored, ε)
    candidates ← REWRITE(layer_front_set)
    depth ← depth + 1
return E-PARETO-FRONT(all scored nodes)
```

A higher ϵ value implies a higher probability to retrieve the Pareto front. But in practice, it was slower and it did not improve significantly the results. In the experiments presented in Section 5.4, we only present the results with PTS that is equivalent to $\mathcal{E}$-PTS with ϵ set to 0.

5.4 Experiments

In this Chapter 5, we have described two search-based approaches for the paraphrase generation task.

The motivation was to overcome the problems encountered by the supervised methods studied in Chapter 4. Indeed, we wanted to elaborate a paraphrase generation strategy that is more generic than the training of encoder-decoder models which are highly biased by their training sets. Using PPDB transformation rules, we can generate a tree of candidate paraphrase from a source sentence. We have developed two search models to explore the generated trees and find the best candidate paraphrase.

Initially, we exploited a Monte-Carlo search and developed a multi-objective score function that combines three axes to be optimized in parallel: semantic similarity, syntax and surface diversity with respect to the source sentence. This is the MCPG model.

To solve the limitations of the MCPG model, we developed the PTS algorithm which approximates the Pareto front obtained with the different scores to be optimized to explore the space of candidate paraphrases. To evaluate these methods, we use CGMH, a baseline presented by Miao et al. [2018] which

has an approach directly comparable to our methods. We also present two baseline search policies *pts-random* and *pts-upperbound* to try to better analyze our results. We present the baselines in more detail in Section 5.4.1.

We detail the setting for the experiments in Section 5.4.2. In section 5.4.3, we detail the results and their analysis. The evaluation metrics and datasets are described respectively in Section 3.2 and 3.3.

5.4.1 Baselines

CGMH

We compare our models with CGMH which is another search-based algorithm presented by Miao et al. [2018]. This baseline has an overall similar search-based approach for paraphrase generation.

Constrained Sentence Generation by Metropolis-Hastings Sampling (CGMH) [Miao et al., 2018] is an approach that uses Metropolis-Hastings sampling [Metropolis et al., 2004] for constrained sentence generation.

Starting from the source sentence, the CGMH algorithm samples a sequence of sentences by using local editions: word replacement, deletion and insertion. For paraphrase generation, CGMH constraints the sentence generation using a matching function that combines a measure of semantic similarity and a measure of English fluency. This model is therefore directly comparable with the MCPG and PTS approaches detailed in Sections 5.2 and 5.3.

In addition, CGMH also includes the semantic diversity axis in the search loop termination. Indeed, as soon as a candidate paraphrase is far enough from the source sentence in terms of BLEU score, the search stops.

This method thus uses a search strategy in a candidate paraphrase space and a paraphrase scoring function comparable to ours. To the best of our knowledge, this is the only published work directly comparable to our models.

To evaluate CGMH as introduced by Miao et al. [2018] we used the code they provided [1]. We managed to reproduce their results on QQP. On our test set, we achieved a BLEUscore of 22.5 while they only reported 18.8. We then extended the experiment to other datasets and metrics. These results are reported in Table 5.3.

PTS baseline and selection policies

As detailed in Section 5.3, the PTS algorithm extracts a subset of paraphrase candidates that are in the Pareto front. The final phase of the algorithm is to select the best paraphrase candidate among the Pareto front. For our experiment, we set the selection policy to optimize the balanced score function developed for MCPG and a new balanced score function optimized a posteriori for the Pareto front. We also used other selection policies to compare to and evaluate the model performances.

We hence developped two selection policies to use as baselines:

— **PTS random**: this policy consists of **sampling randomly** the generated paraphrase among the Pareto front. This naive policy aims at evaluating the quality of the Front.

1. https://github.com/NingMiao/CGMH

— **PTS upper-bound**: the motivation for this policy is to measure the maximum performance that the model could have achieved if it had selected the candidate paraphrase that would have obtained the best score. To be more precise, the policy consists in selecting the paraphrase of the Pareto front that is closest to the reference gold target from the dataset in terms of BLEU score.

Remark While very naive the PTS random selection policy can be used to generate the paraphrases. However, the PTS upper-bound uses the reference paraphrase to generate the best candidate. So the latter can only be used as a reference for analysis purposes to evaluate the actual capacity of the weakly-supervised model to generate a paraphrase that would be close to the reference one.

5.4.2 Settings

Data preprocessing

PPDB rules being lowercase, we lowercased all the source and target sentences from all datasets.

We then proceed to a basic tokenization scheme. We split on whitespaces and considered every punctuation (’:, ’,’, ’!’, ’?’, ’;’) as tokens. Also, again to match the PPDB rules, we split the contractions from the root words. For instance, *"we’ll"* was tokenized as *"we ’ll"* and *"doesn’t"* as *"does n’t"*.

Datasets

We conducted the experiments on the five aligned data corpora presented in Section 3.2 – MSCOCO, QQP, OPUSPARCUS, MSRPARAPHRASE, and PAWS – even though in practice our models are not directly supervised on the data so we do not require aligned corpora. We kept the same train/test splits as for the experiments with the supervised models in Chapter 4.

Evaluation

We evaluated our models and the baselines using four automatic metrics: BLEU, TER, METEOR and BERT. These metrics are described more thoroughly in Section 3.3.

As a reminder, BERT here used as an evaluation metric is the BERT-score computed between the paraphrase produced by the model and the corresponding aligned reference paraphrase in the test data. It measures the semantic similarity between the produced paraphrase and the gold target. We emphasize that this metric is different from the BERT-score used in the score function optimized by the MCPG and PTS models because in this case it compared the source sentence to the candidate paraphrases.

The same reasoning applies to the BLEU metric which is used as a measure of surface diversity by the CGMH model.

5.4.3 Results

Table 5.3 summarizes the results of the comparison of our models MCPG and PTS and the weakly-supervised method CGMH.

Corpus	**Model**	*BLEU* ↑	*TER* ↓	*METEOR* ↑	*BERT* ↑
MSCOCO	CGMH	**17.3**	72.6	21.9	0.7
	MCPG	16.5	73.5	**23.2**	**0.71**
	PTS	17.0	**69.9**	22.8	0.64
	PTS *upper-bound*	17.6	69.8	23.5	0.64
	PTS *random*	6.4	90.1	17.9	0.56
MSRPARA.	CGMH	9.7	72.9	15.4	0.48
	MCPG	39.3	52.4	**37.2**	**0.81**
	PTS	**40.3**	**48.4**	36.1	0.80
	PTS *upper-bound*	49.9	43.0	39.5	0.82
	PTS *random*	28.3	62.9	31.9	0.74
OPUSPAR.	CGMH	7.6	78.9	16.8	0.58
	MCPG	**9.6**	78.6	**23.3**	0.67
	PTS	9.1	**70.1**	22.1	**0.68**
	PTS *upper-bound*	14.5	65.8	24.2	0.68
	PTS *random*	4.3	93.0	18.1	0.57
PAWS	CGMH	15.4	58.1	20.7	0.61
	MCPG	55.5	24.3	**49.2**	**0.93**
	PTS	**57.9**	**21.9**	48.5	0.92
	PTS *upper-bound*	65.3	18.4	51.3	0.93
	PTS *random*	36.5	43.2	41.1	0.82
QQP	CGMH	22.5	65.0	27.0	0.72
	CGMH Miao et al., 2018	18.8	-	-	-
	MCPG	24.1	64.5	**31.8**	**0.78**
	PTS	**25.6**	**58.7**	31.4	**0.78**
	PTS *upper-bound*	32.7	54.2	33.7	0.79
	PTS *random*	9.8	88.4	24.5	0.66

Table 5.3 – **Experiments summary.** Symbol '↑' means that higher value is better. Significantly best values are marked in bold.

Our methods MCPG and PTS outperform the CGMH baseline on all corpora except on the MSCOCO dataset where the results are similar.

Discussion

Comparison with the supervised approach

Table 5.3 summarizes the results of the comparison of our models, supervised encoder-decoder neural networks and the weakly-supervised search-based methods detailed in this Chapter 5.

Overall, these results are mixed. It is however important to keep in mind that contrary to the supervised baselines which are retrained for each dataset, the parameters of the CGMH, MCPG and PTS models are left unchanged.

On the MSCOCO and QQP datasets, the supervised baselines achieve clearly better results, but MCPG and PTS achieve better results on OPUSPARCUS and PAWS except with the BERT score for which the TRANSFORMER model achieves similar results.

These results prove that even without a specialized training sets, generic search-based methods are competitive for paraphrase generation. However, it is a fact that encoder-deoder networks have excellent performances for text generation and have the potential to generate more complex paraphrases than those obtained by simple local transformations as in our models.

Training a general – all-purpose – paraphrase generation network would require a huge volume of data. And there is yet much less aligned corpora available for paraphrase than for translation.

Generic search-based methods can be used as an offline model for data augmentation. In Chapter 6, we conduct experiments on distilling data generated by MCPG and PTS models for TRANSFORMER models.

Chapter 6

Distillate Search Policies

6.1 Background

6.1.1 Supervised Pretraining and Fine-tuning

Pretraining is a strategy used to train a model to solve a specific task that can be complex and hard to optimize. In particular, several pre-training algorithms have been developped to train big neural networks architectures.

Greedy layer-wise supervised training Y. Bengio, Lamblin, et al. [2007] introduce a supervised greedy and layer-wise algorithm to train neural networks. The idea is simple, we train each layer as the hidden layer of a one-hidden layer supervised neural network. After each layer-wise training, the output layer of the network is thrown away and we only keep the trained hidden layer's parameters as pre-training initialization of the new top layer of the neural network. That way, each layers of the neural network is trained iteratively, constructing sequentially the whole architecture.

This algorithm lead to several *greedy algorithms* that break the problem into several subtasks and solve each one separately. Often, greedy algorithms are followed by a *fine-tuning* stage [I. Goodfellow et al., 2016] to ensure a joint optimization of all the subtasks and search for an optimal solution to the full problem.

Transfer learning As discussed by I. Goodfellow et al. [2016], Yosinski et al. [2014] studied supervised pretraining of big neural networks in the context of transfer learning [Y. Bengio, 2012; Y. Bengio, Bastien, et al., 2011; Caruana, 1995]. In transfer learning, a first network is trained on a dataset for some task. Then, the learned features are transfered to a second network that is trained on another dataset for another task. This works if the learned features are general to both tasks.

Usually for transfer learning, a small network is trained on the first task. Then a bigger network is initialized with the learned parameters of the first network and randomly for the new layers. This second network is then trained on the second task. This second task can that way have a smaller training set without overfitting because of the general features learned from the first training.

6.1.2 Distillation

Buciluǎ et al. [2006] compressed the knowledge of an ensemble of models into a single model. Hinton et al. [2015] developed the idea further with the idea of *distillation*. According to Hinton et al. [2015],

a simple form of distillation is when knowledge is transfered to the distilled model by training it on a transfer set. In fact, their work focus on distilling a big neural network into a smaller one. In our case, the knowledge we would like to distill is the results of a learned search policy.

We conducted a simple experiment to try to distil the knowledge learned from the MCPG and PTS search policies introduced in Chapter 5 into a supervised neural network like in Chapter 4.

6.2 Experiments

In Chapter 4, we trained supervised encoder-decoders neural networks for paraphrase generation. The neural networks achieve good performances in many text generation tasks. From our experiments in Section 4.2, we concluded that SEQ2SEQ models could learn to generate complex paraphrase structures, but the main drawback of this method comes from the lack of a large training set. Indeed, the learned models were highly biased by the training data. And in the case of MSRPARAPHRASE, the models performed poorly, probably due to the too small size of the training set.

To overcome the limits of the supervised strategy, in Chapter 5 we studied weakly supervised search-based methods for paraphrase generation. In particular, we developped two search models called MCPG and PTS to generate more generic paraphrases. As seen in Section 5.4, while being weakly supervised, the search approaches achieve comparable results with the supervised methods. In the case of MSRPARAPHRASE, MCPG and PTS outperform the encoder-decoders.

In order to get the best of both worlds, we distill the knowledge learned by MCPG and PTS into an encoder-decoder neural netwotks. One option that can be seen as *transfer learning* is to enrich the training set of a TRANSFORMERwith the results of the search-based methods.

To test this idea, we used our MCPG and PTS models to augment the MSRPARAPHRASE training set.

As presented in Section 3.2, MSRPARAPHRASE is a paraphrase identification corpora. In other words, MSRPARAPHRASE contains paraphrase pairs and non-paraphrase pairs. For our paraphrase generation experiments, we only needed the paraphrase pairs, so we threw away the other pairs. However, the non-paraphrase pairs belong to the same data distribution as the paraphrase pairs used to train the supervised encoder-decoder on MSRPARAPHRASE. We make use of these non-paraphrase pairs unused for the TRANSFORMER training in Section 4.2 for our distillation experiment.

6.2.1 Settings

First, we turn the non-paraphrase pairs of MSRPARAPHRASE into source sentences to be paraphrased by the models we want to distil namely MCPG and PTS.

Then, we swap the generated paraphrase with the source sentence to create new paraphrase pairs. The generated paraphrases become the new source sentences. And the initial source sentences become the new reference sentences. This technique, called *back-translation*, was introduced by Sennrich et al. [2016] and Edunov et al. [2018]. It ensures that the model always has a syntactically correct sentence as output reference. Indeed, as an encoder-decoder is learned by teacher forcing on the reference sentence, learning with a noisy reference can degrade the training.

With these newly generated paraphrase pairs, we augmented the MSRPARAPHRASE training set. We then trained new supervised TRANSFORMER models on the augmented training sets, one augmented with MCPG and the other one augmented with PTS.

6.2.2 Results

Train set	*BLEU* ↑	*TER* ↓	*METEOR* ↑	*BERT* ↑
ORIG.	20.7	76.7	21.6	**0.65**
ORIG. + MCPGPARAPHRASES	**25.3**	**69.6**	24.5	0.63
ORIG. + PTSPARAPHRASES	25.3	71.1	**24.6**	0.63

Table 6.1 – **Data-augmentation experiment summary**. Symbol '↑' means that higher value is better. Best values are marked in bold. We trained a TRANSFORMER base model on three versions of the MSRPARAPHRASE dataset: the original training set (ORIG.) and the original training set extended by paraphrasing other sentences from the same distribution with the MCPG and PTS models. The models trained with the extended datasets achieve better results. This shows that our weakly-supervised models can be distilled to improve state-of-the-art models.

We report the results of this experiment in Table 6.1. The models trained with the augmented training set achieved a significant performance gain on BLEU, TER and METEOR.

6.2.3 Analysis

State-of-the-art of paraphrase generation is achieved by encoder-decoder models that have the ability to learn complex sentence transformations. For paraphrase generation, these methods are limited by the biased and small training datasets. By augmenting the training datasets with paraphrase pairs generated by search-based models, we are able to improve the neural networks without overfitting.

As shown in Table 6.1, augmenting the training set of MSRPARAPHRASE by paraphrase pairs generated by MCPG and PTS increased the BLEU score by 5 points. On the contrary, the last column of the table also shows that the BERT score has slightly decreased. As illustrated in Table 5.1 showing candidate paraphrases examples, the candidates selected by the search policy obtains a BERT scores with respect to the source equal to 0.96. Additionnaly, the MCPG model on its own achieves a BERT score performance of 0.81 (cf. Table 5.3).

Despite these surprising results with the BERT score, the search-based approaches seem to be very interesting as offline data augmentation systems to overcome the problems of the encoder-decoders. It provides a compromise between supervised and weakly-supervised approaches.

6.3 Conclusion on Paraphrase Generation

In this second part of this manuscript, we presented the research work on paraphrase generation.

The paraphrase generation task presented in Chapter 3 is difficult. The structure we model is the structure of the language itself and paraphrase generation is far from being solved.

Supervised encoder-decoder models, which are very popular for all text generation tasks are not yet as efficient as for the translation task, for example.

In fact, when we started this work, the state-of-the-art was achieved by LSTM or Residual-LSTM models [Prakash, Sadid A. Hasan, et al., 2016a]. Later, the TRANSFORMER model [Vaswani, Shazeer, Parmar, Uszkoreit, L. Jones, Aidan N. Gomez, et al., 2017a], as for many other tasks, obtained very good results for the generation of paraphrases [Egonmwan and Chali, 2019a]. The state-of-the-art is evolving very fast and a lot of research have been conducted so far.

In Chapter 4, we conducted experiments on using supervised encoder-decoder models for the task of paraphrase generation. Additionnaly, we have homogenised the experiments presented in the literature, which were difficult to reproduce and not comparable. From these experiments, we extracted the main limitation to this type of model: the lack of a large training dataset. Indeed, the learned models are too biased, and tend to overfit when the training set is too small.

This lead us to explore another approach for paraphrase generation: the search-based approach. In Chapter 5, we defined a framework of paraphrase generation as a search problem in a lattice of paraphrase candidates. To find the best paraphrase candidate in this vast space we studied two search algorithms.

First, MCPG implements an MCTS approach with a multi-objective reward function to find the best candidate. The scoring function we used combines three dimensions: the semantic similarity, the syntax correctness and the surface diversity.

We oberved two drawbacks for MCTS. It was designed for combinatorial problems like Go where the evaluation is only possible on the leaves of the search tree. This is not the case for paraphrase generation where the neural models can evaluate any rewriting step and where rewriting from good candidates is more likely to provide good paraphrases than rewriting from bad ones. Also , it has been designed for single-criterion search which requires fixing the balances between criteria definitively before any paraphrase search begins. This is not very flexible, and it becomes painful when we want to generate sets of good candidates.

To overcome these issues, we developped another search strategy that approximates the Pareto front between the three criteria. This is the PTS model.

These seach-based models outperform a baseline from the literature CGMH and achieve performances comparable with the supervised neural networks' ones.

However, search-based models are cumbersome and not at all suitable for fast generation. To find the best of both worlds, in Chapter 6 we conducted an experiment to distill MCPG and PTS knowledge into TRANSFORMER models. The distillation improved the performances of the TRANSFORMER model on MSRParaphrase and is a promising research branch.

Publications

This work has been published at conferences in two parts:

— the search-based methods presented in Chapter 5 were published at a workshop on learning with combinatorial algorithms :

◇ **Neural-Driven Multi-criteria Tree Search for Paraphrase Generation.**
Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D. (2020, December). *In Learning Meets Combinatorial Algorithms (LMCA) Workshop at NeurIPS 2020.*

— the supervised models and the distillation experiment presented respectively in Chapters 4 and 6 were published as a conference article at EACL :

◇ **Neural-Driven Search-Based Paraphrase Generation.**
Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D. (2021, April). *In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (pp. 2100-2111).*

Conclusion

Conclusion

The encoder-decoder model is at the heart of the work presented in this thesis. This family of neural networks built for sequence to sequence tasks are well suited for text generation tasks like data-to-text or paraphrase generation and have improved the state-of-the-art in many text generation tasks. The Transformer model performances lead to the development of pre-trained transformer-based models like BERT contextual embeddings and the GPT-2 language model. The Transformer is also widely used as a supervised model learned from scratch for text generation tasks on aligned corpora and often outperforms the original Seq2Seq that relies on RNNs.

Following this idea, in Chapter 2, we explored the data-to-text task of RDF verbalization. We trained supervised Transformer models on a newly released version of the WebNLG dataset and studied in depth several pre-training strategies to overcome the small size of the aligned corpus. This first experiment (see Section 2.4.3) on text generation with supervised models highlighted the limit of such models with a lack of generalization.

The field of paraphrase generation is closely related to machine translation due to the nature of the task. However, the latter received much more attention because of the availabily of massive aligned corpora. For paraphrase generation, the corpora are smaller and more biased (See Section 3.2).

In this thesis, we have trained Transformer models on aligned corpora to directly compare with the litterature model that was based on RNNs at the time. This lead to two issues. First, we found it very difficult to reproduce the state-of-the-art results for the paraphrase generation task. An important contribution of the thesis was to propose a uniform experimental framework for comparing encoder-decoder models for paraphrase generation and to extend the training to 5 aligned paraphrase corpora including the two highly biased ones – MSCOCO and QQP – often used in the literature. Second, the Transformer models did not achieve the expected results. Indeed, in Section 4.2, we reported no significant differences between the RNN-based neural networks architectures and the Transformers except on MSRParaphrase. We assume that as MSRParaphrase is smaller and contains longer sentences, then the Transformer overfit less than the RNN-based neural networks.

Due to the nature of the task and data available for supervised training, we followed the path of search-based alternative strategies to generate paraphrases. The main motivation was to provide a better control of the generated paraphrase. In Chapter 5, we described a search-based scheme where candidate paraphrases are generated by iterative rewritting of the source sentence. For that, we casted the paraphrase generation tasks as a tree-search problem. We then, developped two search strategies MCPG and PTS and a paraphrase scoring module that leverages the BERTscore, GPT-2 and the Levenshtein distance. The results reported in Section 5.4 showed that without a specialized training set, generic search-based methods are competitive for paraphrase generation but it is clear that the supervised models have the potential to generate more complex paraphrases.

Finally, as the search-based methods could only be used as an offline model for data augmentation,

we conducted experiments of data distillation for the TRANSFORMER model in Chapter 6. The experiments showed that the distillation improved the performances of the TRANSFORMER model trained on MSRPARAPHRASE and is a promising research branch.

In order to dig deeper into this idea of pre-training and distillation. An extension to this work could be to fine-tune a general pre-trained language model to the task of paraphrase generation. We hypothesis that this would tackle the issue of few and biased data that we encountered when training transformer models in chapter 4.

Another perspective to this work would be to develop new search policies in the search-based framework. In particular, one could think of a fully learned policy with reinforcement learning. The main difficulty is to find a good oracle to train the policy. We assume this would be difficult but an interesting research avenue.

Also, one could add more control to the supervised scheme by exploring the structured prediction solutions to text generation.

Publications

Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D., (2020), Neural-Driven Multi-criteria Tree Search for Paraphrase Generation, *in Learning Meets Combinatorial Algorithms (LMCA) Workshop at NeurIPS 2020.*

Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D., (2021), Neural-Driven Search-Based Paraphrase Generation, *in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume.*

Montella, S., Fabre, B., Urvoy, T., Heinecke, J., & Rojas-Barahona, L., (2020), Denoising Pre-Training and Data Augmentation Strategies for Enhanced RDF Verbalization with Transformers, *arXiv preprint arXiv:2012.00571.*

References

Auer, P., Cesa-Bianchi, N., & Fischer, P., (2002), Finite-time Analysis of the Multiarmed Bandit Problem, *Mach. Learn.*, *472-3*, 235–256, https://doi.org/10.1023/A:1013689704352

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., & Bengio, Y., (2016), An Actor-Critic Algorithm for Sequence Prediction.

Bahdanau, D., Cho, K., & Bengio, Y., (2016), Neural Machine Translation by Jointly Learning to Align and Translate [arXiv: 1409.0473], *arXiv:1409.0473 [cs, stat]*, retrieved June 29, 2020, from http://arxiv.org/abs/1409.0473
Comment: Accepted at ICLR 2015 as oral presentation

Banerjee, S., & Lavie, A., (2005), METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, *in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan, Association for Computational Linguistics, retrieved November 15, 2019, from https://www.aclweb.org/anthology/W05-0909

Bannard, C., & Callison-Burch, C., (2005), Paraphrasing with bilingual parallel corpora, *in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., & Zampieri, M., (2019), Findings of the 2019 Conference on Machine Translation (WMT19), *in Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, Florence, Italy, Association for Computational Linguistics, https://doi.org/10.18653/v1/W19-5301

Barzilay, R., & McKeown, K. R., (2001), Extracting paraphrases from a parallel corpus, *in Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*, Toulouse, France, Association for Computational Linguistics, https://doi.org/10.3115/1073012.1073020

Beeferman, D., & Berger, A., (2000), Agglomerative clustering of a search engine query log, *in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N., (2015), Scheduled sampling for sequence prediction with recurrent neural networks, *Advances in neural information processing systems*, *28*.

Bengio, Y., (2012), Deep learning of representations for unsupervised and transfer learning, *in Proceedings of ICML workshop on unsupervised and transfer learning*, JMLR Workshop and Conference Proceedings.

Bengio, Y., Bastien, F., Bergeron, A., Boulanger–Lewandowski, N., Breuel, T., Chherawala, Y., Cisse, M., Côté, M., Erhan, D., Eustache, J. Et al., (2011), Deep learners benefit more from out-of-distribution examples, *in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H., (2007), Greedy layer-wise training of deep networks, *in Advances in neural information processing systems*.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J., (2009), Curriculum Learning, *in Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Quebec, Canada, Association for Computing Machinery, https://doi.org/10.1145/1553374.1553380

Bengio, Y., Simard, P., & Frasconi, P., (1994), Learning long-term dependencies with gradient descent is difficult, *IEEE transactions on neural networks*, *5 2*, 157–166.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T., (2017), Enriching Word Vectors with Subword Information, *Transactions of the Association for Computational Linguistics*, *5*, 135–146, https://doi.org/10.1162/tacl_a_00051

Bojar, O., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Koehn, P., & Monz, C., (2018), Findings of the 2018 Conference on Machine Translation (WMT18), *in Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, Belgium, Brussels, Association for Computational Linguistics, https://doi.org/10.18653/v1/W18-6401

Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Mercer, R., & Roossin, P., (1988), A Statistical Approach to Language Translation, *in Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, Budapest, Hungry,

Association for Computational Linguistics, https://doi.org/10.3115/991635.991651

Brown, P. E., Pietra, V. J. D., Pietra, S. A. D., & Mercer, R. L., (1993), The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics*, *19 2*, 50.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Roossin, P. S., (1990), A Statistical Approach to Machine Translation, *Computational Linguistics*, *16 2*, 79–85, https://aclanthology.org/J90-2002

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A., (2006), Model compression, *in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Callison-Burch, C., (2008), Syntactic Constraints on Paraphrases Extracted from Parallel Corpora, *in Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii, Association for Computational Linguistics, retrieved October 21, 2019, from https://www.aclweb.org/anthology/D08-1021

Callison-Burch, C., Koehn, P., & Osborne, M., (2006a), Improved Statistical Machine Translation Using Paraphrases, *in Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA, Association for Computational Linguistics, retrieved October 28, 2019, from https://www.aclweb.org/anthology/N06-1003

Callison-Burch, C., Koehn, P., & Osborne, M., (2006b), Improved statistical machine translation using paraphrases, *in Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

Cao, Z., Luo, C., Li, W., & Li, S., (2017), Joint Copying and Restricted Generation for Paraphrase, *in Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14527

Carroll, J., Minnen, G., Pearce, D., Canning, Y., Devlin, S., & Tait, J., (1999), Simplifying Text for Language-Impaired Readers, *in Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, Association for Computational Linguistics, https://www.aclweb.org/anthology/E99-1042

Caruana, R., (1995), Learning many related tasks at the same time with backpropagation, *in Advances in neural information processing systems*.

Castro Ferreira, T., Gardent, C., Ilinykh, N., van der Lee, C., Mille, S., Moussallem, D., & Shimorina, A., (2020), The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020), *in Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), Association for Computational Linguistics, https://aclanthology.org/2020.webnlg-1.7

Castro-Ferreira, T., Gardent, C., Ilinykh, N., van der Lee, C., Mille, S., Moussalem, D., & Shimorina, A., (2020), The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020), *in Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual), Association for Computational Linguistics.

Chandrasekar, R., & Srinivas, B., (1997), Automatic induction of rules for text simplification, *Knowledge-Based Systems*, *10 3*, 183–190, https://doi.org/10.1016/S0950-7051(97)00029-4

Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daumé III, H., & Langford, J., (2015), Learning to search better than your teacher, *in International Conference on Machine Learning*, PMLR.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollar, P., & Zitnick, C. L., (2015), Microsoft COCO Captions: Data Collection and Evaluation Server [arXiv: 1504.00325], *arXiv:1504.00325 [cs]*, retrieved June 17, 2019, from http://arxiv.org/abs/1504.00325

Comment: arXiv admin note: text overlap with arXiv:1411.4952

Chevelu, J., Lavergne, T., Lepage, Y., & Moudenc, T., (2009), Introduction of a new paraphrase generation tool based on Monte-Carlo sampling, *in Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, Suntec, Singapore, Association for Computational Linguistics, retrieved October 21, 2019, from https://www.aclweb.org/anthology/P09-2063

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y., (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078*.

Cimiano, P., Lüker, J., Nagel, D., & Unger, C., (2013), Exploiting ontology lexica for generating natural language texts from RDF data, *in Proceedings of the 14th European Workshop on Natural Language Generation*.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D., (2020), ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, *in 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, https://openreview.net/forum?id=r1xMH1BtvB

Consortium, W. W. W. Et al., (2014), RDF 1.1 concepts and abstract syntax.

Creutz, M., (2018), Open Subtitles Paraphrase Corpus for Six Languages, *in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, European Language Resources Association (ELRA), retrieved November 15, 2019, from https://www.aclweb.org/anthology/L18-1218

Culicover, P. W., (1968), Paraphrase generation and information retrieval from stored text., *Mech. Transl. Comput. Linguistics*, *11 3-4*, 78–88.

Daumé, H., Langford, J., & Marcu, D., (2009), Search-based structured prediction, *Machine Learning*, *75 3*, 297–325, https://doi.org/10.1007/s10994-009-5106-x

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. N., (2018), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, https://arxiv.org/abs/1810.04805

Dolan, B., & Brockett, C., (2005), Automatically Constructing a Corpus of Sentential Paraphrases, *in Third International Workshop on Paraphrasing (IWP2005)*, Asia Federation of Natural Language Processing, https://www.microsoft.com/en-us/research/publication/automatically-constructing-a-corpus-of-sentential-paraphrases/

Dras, M., (1999), *Tree adjoining grammar and the reluctant paraphrasing of text*, Macquarie University Sydney.

Duma, D., & Klein, E., (2013), Generating natural language from linked data: Unsupervised template extraction, *in Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*.

Edunov, S., Ott, M., Auli, M., & Grangier, D., (2018), Understanding Back-Translation at Scale, *in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, Association for Computational Linguistics, https://doi.org/10.18653/v1/D18-1045

Egonmwan, E., & Chali, Y., (2019a), Transformer and seq2seq model for Paraphrase Generation, *in Proceedings of the 3rd Workshop on Neural Generation and Translation*, Hong Kong, Association for Computational Linguistics, https://doi.org/10.18653/v1/D19-5627

Egonmwan, E., & Chali, Y., (2019b), Transformer and seq2seq model for Paraphrase Generation, *in Proceedings of the 3rd Workshop on Neural Generation and Translation*, Hong Kong, Association for Computational Linguistics, https://doi.org/10.18653/v1/D19-5627

Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D., (2020), Neural-Driven Multi-criteria Tree Search for Paraphrase Generation, *in Learning Meets Combinatorial Algorithms (LMCA) Workshop at NeurIPS 2020*.

Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D., (2021), Neural-Driven Search-Based Paraphrase Generation, *in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Ferreira, T. C., van der Lee, C., Krahmer, E., & Wubben, S., (2017), Tilburg University models for the WebNLG challenge, *in International Conference on Natural Language Generation 2017*.

Freitag, M., Grangier, D., & Caswell, I., (2020), BLEU might be guilty but references are not innocent, *arXiv preprint arXiv:2004.06063*.

Fu, Y., Feng, Y., & Cunningham, J. P., (2020), Paraphrase Generation with Latent Bag of Words [arXiv: 2001.01941], *arXiv:2001.01941 [cs]*, retrieved February 17, 2020, from http://arxiv.org/abs/2001.01941
Comment: NeurIPS 19 camera ready

Gammerman, A., Vovk, V., & Vapnik, V., (1998a), Learning by Transduction, *in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, Morgan Kaufmann Publishers Inc.

Gammerman, A., Vovk, V., & Vapnik, V., (1998b), Learning by Transduction, *in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, Morgan Kaufmann Publishers Inc.

Ganitkevitch, J., Van Durme, B., & Callison-Burch, C., (2013), PPDB: The Paraphrase Database, *in Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, Association for Computational Linguistics, retrieved November 27, 2019, from https://www.aclweb.org/anthology/N13-1092

Gao, H., Wu, L., Hu, P., & Xu, F., (2020), RDF-to-Text Generation with Graph-augmented Structural Neural Encoders (C. Bessiere, Ed.) [Main track], *in* C. Bessiere (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, International Joint Conferences on Artificial Intelligence Organization, Main track, https://doi.org/10.24963/ijcai.2020/419

Gao, S., Zhang, Y., Ou, Z., & Yu, Z., (2020), Paraphrase augmented task-oriented dialog generation, *arXiv preprint arXiv:2004.07462*.

Gardent, C., & Barahona, L. M. R., (2013), Using paraphrases and lexical semantics to improve the accuracy and the robustness of supervised models in situated dialogue systems, *in Conference on Empirical Methods in Natural Language Processing*.

Gardent, C., Shimorina, A., Narayan, S., & Perez-Beltrachini, L., (2017a), Creating Training Corpora for NLG Micro-Planners, *in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, Association for Computational Linguistics, https://doi.org/10.18653/v1/P17-1017

Gardent, C., Shimorina, A., Narayan, S., & Perez-Beltrachini, L., (2017b), The WebNLG Challenge: Generating Text from RDF Data, *in Proceedings of the 10th International Conference on Natural Language Generation*, Santiago de Compostela, Spain, Association for Computational Linguistics, https://doi.org/10.18653/v1/W17-3518

Gelly, S., & Silver, D., (2007), Combining Online and Offline Knowledge in UCT, retrieved November 28, 2019, from https://hal.inria.fr/inria-00164003

Goodfellow, I., Bengio, Y., & Courville, A., (2016), *Deep learning*, MIT press.

Goodfellow, I. J., Shlens, J., & Szegedy, C., (2014), Explaining and harnessing adversarial examples, *arXiv preprint arXiv:1412.6572*.

Guo, Q., Jin, Z., Qiu, X., Zhang, W., Wipf, D., & Zhang, Z., (2020), CycleGT: Unsupervised Graph-to-Text and Text-to-Graph Generation via Cycle Training, *ArXiv*, *abs/2006.04702*.

Gupta, A., Agarwal, A., Singh, P., & Rai, P., (2017), A Deep Generative Framework for Paraphrase Generation [arXiv: 1709.05074], *arXiv:1709.05074 [cs]*, retrieved September 11, 2019, from http://arxiv.org/abs/1709.05074

Gupta, A., Agarwal, A., Singh, P., & Rai, P., (2018), A Deep Generative Framework for Paraphrase Generation, *in Thirty-Second AAAI Conference on Artificial Intelli-*

gence, retrieved November 29, 2019, from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16353

Hinton, G., Vinyals, O., & Dean, J., (2015), Distilling the knowledge in a neural network, *arXiv preprint arXiv:1503.02531*.

Hochreiter, S., & Schmidhuber, J., (1997), Long short-term memory, *Neural computation*, *9 8*, 1735–1780.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S. Et al., (2021), Knowledge graphs, *ACM Computing Surveys (CSUR)*, *54 4*, 1–37.

Iso, H., Qiao, C., & Li, H., (2020), Fact-based Text Editing, *in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Association for Computational Linguistics, https://doi.org/10.18653/v1/2020.acl-main.17

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L., (2018), Adversarial example generation with syntactically controlled paraphrase networks, *arXiv preprint arXiv:1804.06059*.

Jarke, M., Neumann, B., Vassiliou, Y., & Wahlster, W., (1989), KBMS requirements of knowledge-based systems. *in Foundations of knowledge base management* (pp. 381–394), Springer.

Jones, R., Rey, B., Madani, O., & Greiner, W., (2006), Generating Query Substitutions, *in Proceedings of the 15th International Conference on World Wide Web*, Edinburgh, Scotland, Association for Computing Machinery, https://doi.org/10.1145/1135777.1135835

Kocsis, L., & Szepesvári, C., (2006), Bandit Based Monte-carlo Planning [event-place: Berlin, Germany], *in Proceedings of the 17th European Conference on Machine Learning*, Berlin, Heidelberg, Springer-Verlag, event-place: Berlin, Germany, https://doi.org/10.1007/11871842_29

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R. Et al., (2007), Moses: Open source toolkit for statistical machine translation, *in Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*.

Koehn, P., Och, F. J., & Marcu, D., (2003), Statistical Phrase-Based Translation, *in Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, retrieved October 22, 2019, from https://www.aclweb.org/anthology/N03-1017

Kornél Csernai, (2017), First Quora Dataset Release: Question Pairs - Data @ Quora - Quora, retrieved November 14, 2019, from https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Lavie, A., Sagae, K., & Jayaraman, S., (2004), The significance of recall in automatic metrics for MT evaluation, *in Conference of the Association for Machine Translation in the Americas*, Springer.

Leblond, R., Alayrac, J.-B., Osokin, A., & Lacoste-Julien, S., (2018), SEARNN: Training RNNs with global-local losses, *in International Conference on Learning Representations*.

Legriel, J., Le Guernic, C., Cotton, S., & Maler, O., (2010), Approximating the pareto front of multi-criteria optimization problems, *in International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S. Et al., (2015), Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, *Semantic web*, *6 2*, 167–195.

Levenshtein, V. I., (1965), Binary codes capable of correcting deletions, insertions, and reversals, *in Doklady Akademii Nauk*, Russian Academy of Sciences.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. [Ves], & Zettlemoyer, L., (n.d.), BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. [Veselin], & Zettlemoyer, L., (2020), BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, *in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Association for Computational Linguistics, https://doi.org/10.18653/v1/2020.acl-main.703

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L., (2014), Microsoft COCO: Common Objects in Context, *in Computer Vision – ECCV 2014*, Springer, Cham, https://doi.org/10.1007/978-3-319-10602-1_48

Lison, P., & Tiedemann, J., (2016), Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., & Zaremba, W., (2015), Addressing the Rare Word Problem in Neural Machine Translation, *in Proceedings of the 53rd Annual*

Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, Association for Computational Linguistics, https://doi.org/10.3115/v1/P15-1002
First NTM system that works better than its phrase-based counter-part. At WMT'14 contest

Madnani, N., & Dorr, B. J., (2010), Generating Phrasal and Sentential Paraphrases: A Survey of Data-Driven Methods, *Computational Linguistics*, *36 3*, 341–387, https://doi.org/10.1162/coli_a_00002

Marcheggiani, D., & Perez-Beltrachini, L., (2018), Deep Graph Convolutional Encoders for Structured Data to Text Generation, *in Proceedings of the 11th International Conference on Natural Language Generation*.

Mareček, D., & Rosa, R., (2019), From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions, *in Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

McKeown, K. R., (1979), Paraphrasing Using Given and New Information in a Question-Answer System, *in 17th Annual Meeting of the Association for Computational Linguistics*, La Jolla, California, USA, Association for Computational Linguistics, https://doi.org/10.3115/982163.982182

Meteer, M., & Shaked, V., (1988), STRATEGIES FOR EFFECTIVE PARAPHRASING, *in Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*, retrieved November 25, 2019, from https://www.aclweb.org/anthology/C88-2088

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E., (2004), Equation of State Calculations by Fast Computing Machines [Publisher: American Institute of PhysicsAIP], *The Journal of Chemical Physics*, *21 6*, 1087, https://doi.org/10.1063/1.1699114

Metzler, D., Dumais, S., & Meek, C., (2007), Similarity measures for short segments of text, *in European conference on information retrieval*, Springer.

Miao, N., Zhou, H., Mou, L., Yan, R., & Li, L., (2018), CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling [arXiv: 1811.10996], *arXiv:1811.10996 [cs, math, stat]*, retrieved March 9, 2020, from http://arxiv.org/abs/1811.10996
Comment: AAAI19

Mille, S., & Dasiopoulou, S., (2017), Forge at webnlg 2017.

Montella, S., Fabre, B., Urvoy, T., Heinecke, J., & Rojas-Barahona, L., (2020), Denoising Pre-Training and Data Augmentation Strategies for Enhanced RDF Verbalization with Transformers, *arXiv preprint arXiv:2012.00571*.

Moussallem, D., Gnaneshwar, D., Castro Ferreira, T., & Ngonga Ngomo, A.-C., (2020), NABU - Multilingual Graph-based Neural RDF Verbalizer, *in The Semantic Web – ISWC 2020*, Springer International Publishing.

Naber, D., (2003), LanguageTool: A Rule-Based Style and Grammar Checker, *University of Bielefeld*, https://languagetool.org

Olah, C., (2015), Understanding lstm networks.

Osborne, M. J., & Rubinstein, A., (1994), *A course in game theory*, MIT press.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., & Auli, M., (2019), fairseq: A Fast, Extensible Toolkit for Sequence Modeling, *in Proceedings of NAACL-HLT 2019: Demonstrations*.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J., (2001), BLEU: a method for automatic evaluation of machine translation, *in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, Philadelphia, Pennsylvania, Association for Computational Linguistics, https://doi.org/10.3115/1073083.1073135

Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., & Callison-Burch, C., (2015), PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification, *in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China, Association for Computational Linguistics, https://doi.org/10.3115/v1/P15-2070

Pennington, J., Socher, R., & Manning, C. D., (2014), Glove: Global vectors for word representation, *in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Peters, M. E., Ruder, S., & Smith, N. A., (2019), To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks, *in Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, Florence, Italy, Association for Computational Linguistics, https://doi.org/10.18653/v1/W19-4302

Phong, N. T., & Dang, T. N., (2017), https://webnlg-challenge.loria.fr/files/uit-vnu-hcm_report.pdf

Popović, M., (2016), chrF deconstructed: beta parameters and n-gram weights, *in Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*.

Popović, M., (2017), chrF++: words helping character n-grams, *in Proceedings of the second conference on machine translation*.

Post, M., (2018), A Call for Clarity in Reporting BLEU Scores, *in Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels, Association for Computational Linguistics, https://doi.org/10.18653/v1/W18-6319

Prakash, A., Hasan, S. A. [Sadid A.], Lee, K., Datla, V., Qadir, A., Liu, J., & Farri, O., (2016a), Neural Paraphrase Generation with Stacked Residual LSTM Networks, *in Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, The COLING 2016 Organizing Committee, https://www.aclweb.org/anthology/C16-1275

Prakash, A., Hasan, S. A. [Sadid A], Lee, K., Datla, V., Qadir, A., Liu, J., & Farri, O., (2016b), Neural Paraphrase Generation with Stacked Residual LSTM Networks, 12.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I., (2019), Language models are unsupervised multitask learners, *OpenAI Blog*, *1 8*, 9.

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W., (2015), Sequence Level Training with Recurrent Neural Networks [arXiv: 1511.06732], *arXiv:1511.06732 [cs]*, retrieved September 10, 2019, from http://arxiv.org/abs/1511.06732

Reiter, E., & Dale, R., (1997), Building Applied Natural Language Generation Systems, *Nat. Lang. Eng.*, *3 1*, 57–87, https://doi.org/10.1017/S1351324997001502

Ross, S., Gordon, G., & Bagnell, D., (2011), A reduction of imitation learning and structured prediction to no-regret online learning, *in Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J., (1986), Learning representations by back-propagating errors, *nature*, *323 6088*, 533–536.

Sabour, S., Chan, W., & Norouzi, M., (2018), Optimal Completion Distillation for Sequence Learning, *in International Conference on Learning Representations*.

Sahami, M., & Heilman, T. D., (2006), A Web-Based Kernel Function for Measuring the Similarity of Short Text Snippets, *in Proceedings of the 15th International*

Conference on World Wide Web, Edinburgh, Scotland, Association for Computing Machinery, https://doi.org/10.1145/1135777.1135834

Schmidt, F., (2019), Generalization in generation: A closer look at exposure bias, *arXiv preprint arXiv:1910.00292*.

See, A., Liu, P., & Manning, C., (2017), Get To The Point: Summarization with Pointer-Generator Networks, *in Association for Computational Linguistics*, https://arxiv.org/abs/1704.04368

Sekine, S., (2006), On-demand information extraction, *in Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Sellam, T., Das, D., & Parikh, A. P., (2020), BLEURT: Learning robust metrics for text generation, *arXiv preprint arXiv:2004.04696*.

Sennrich, R., Haddow, B., & Birch, A., (2016), Improving Neural Machine Translation Models with Monolingual Data, *in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Association for Computational Linguistics, https://doi.org/10.18653/v1/P16-1009

Shi, X., & Yang, C. C., (2007), Mining related queries from web search engine query logs using an improved association rule mining model, *Journal of the American Society for Information Science and Technology*, *58 12*, 1871–1883.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. v. d., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D., (2016), Mastering the game of Go with deep neural networks and tree search, *Nature*, *529 7587*, 484–489, https://doi.org/10.1038/nature16961

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J., (2006), A Study of Translation Edit Rate with Targeted Human Annotation, *in Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, Cambridge, Massachusetts, USA, Association for Machine Translation in the Americas, https://aclanthology.org/2006.amta-papers.25

Sun, X., & Mellish, C. [Chris], (2007), An experiment on free generation from single RDF triples, *Proceedings of the 11th European Workshop on Natural Language Generation, ENLG 07*, 105–108, https://doi.org/10.3115/1610163.1610181

Sun, X., & Mellish, C. [Christopher Stuart], (2006), Domain Independent Sentence Generation from RDF Representations for the Semantic Web, *in ECAI06 Combined Workshop on Language-Enabled Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems*.

Sutskever, I., Vinyals, O., & Le, Q. V., (2014), Sequence to sequence learning with neural networks, *in Advances in neural information processing systems*.

Tianyi, (2019), Tiiiger/bert_score [original-date: 2019-04-20T15:04:58Z], retrieved September 4, 2019, from https://github.com/Tiiiger/bert_score

Tomas Mikolov, (2012), *Statistical Language Models Based on Neural Networks* (Doctoral dissertation), FIT VUT v Brne, Brno, retrieved January 14, 2020, from https://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf

Trisedya, B. D., Qi, J., Zhang, R., & Wang, W., (2018), GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data, *in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, Association for Computational Linguistics, https://doi.org/10.18653/v1/P18-1151

Tseng, B.-H., Cheng, J., Fang, Y., & Vandyke, D., (2020), A Generative Model for Joint Natural Language Understanding and Generation, *in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Association for Computational Linguistics, https://doi.org/10.18653/v1/2020.acl-main.163

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, Ł., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N., & Uszkoreit, J., (2018), Tensor2Tensor for Neural Machine Translation [arXiv: 1803.07416], *arXiv:1803.07416 [cs, stat]*, retrieved November 29, 2019, from http://arxiv.org/abs/1803.07416

Comment: arXiv admin note: text overlap with arXiv:1706.03762

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. [Aidan N.], Kaiser, L., & Polosukhin, I., (2017a), Attention Is All You Need [arXiv: 1706.03762], *arXiv:1706.03762 [cs]*, retrieved January 24, 2019, from http://arxiv.org/abs/1706.03762

Comment: 15 pages, 5 figures

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. [Aidan N], Kaiser, Ł., & Polosukhin, I., (2017b), 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 (pp. 5998–6008), Curran Associates, Inc., http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Vrandečić, D., & Krötzsch, M., (2014), Wikidata: a free collaborative knowledgebase, *Communications of the ACM*, *5710*, 78–85.

Weaver, W. Et al., (1955), Translation, *Machine translation of languages*, *14 15-23*, 10.

Welleck, S., Brantley, K., Iii, H. D., & Cho, K., (2019), Non-monotonic sequential text generation, *in International Conference on Machine Learning*, PMLR.

Xu, W., Napoles, C., Pavlick, E., Chen, Q., & Callison-Burch, C., (2016), Optimizing Statistical Machine Translation for Text Simplification, *Transactions of the Association for Computational Linguistics*, *4*, 401–415, https://doi.org/10.1162/tacl_a_00107

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V., (2019), XLNet: Generalized Autoregressive Pretraining for Language Understanding, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 5753–5763), Curran Associates, Inc., http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H., (2014), How transferable are features in deep neural networks?, *arXiv preprint arXiv:1411.1792*.

Zhang, B., (2016), Google's neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint arXiv:1609.08144*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y., (2019), BERTScore: Evaluating Text Generation with BERT [arXiv: 1904.09675], *arXiv:1904.09675 [cs]*, retrieved September 4, 2019, from http://arxiv.org/abs/1904.09675
Comment: Code available at https://github.com/Tiiiger/bert_score

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., & Artzi, Y., (2020), BERTScore: Evaluating Text Generation with BERT, *in International Conference on Learning Representations*, https://openreview.net/forum?id=SkeHuCVFDr

Zhang, Y., Baldridge, J., & He, L., (2019), PAWS: Paraphrase Adversaries from Word Scrambling, *in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Association for Computational Linguistics, https://doi.org/10.18653/v1/N19-1131

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q., (2019), ERNIE: Enhanced Language Representation with Informative Entities, *in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zhao, C., Walker, M., & Chaturvedi, S., (2020), Bridging the Structural Gap Between Encoding and Decoding for Data-To-Text Generation, *in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Association for Computational Linguistics, https://doi.org/10.18653/v1/2020.acl-main.224

Zhao, J., Huang, F., Lv, J., Duan, Y., Qin, Z., Li, G., & Tian, G., (2020), Do RNN and LSTM have Long Memory? (H. D. III & A. Singh, Eds.), *in* H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, PMLR, https://proceedings.mlr.press/v119/zhao20c.html

Zhu, J., Park, T., Isola, P., & Efros, A. A., (2017), Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, *in 2017 IEEE International Conference on Computer Vision (ICCV)*.

Zhu, Y., Wan, J., Zhou, Z., Chen, L., Qiu, L., Zhang, W., Jiang, X., & Yu, Y., (2019), Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence, *in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Paris, France, Association for Computing Machinery, https://doi.org/10.1145/3331184.3331232

Appendices

Appendix A

Paraphrase scoring module

In this appendix, we detail the development and the implementation of the scoring module used for the MCPG model presented in Chapter 5 and in particular in section 5.2.

The scoring module that scores the pairs of paraphrases generated by MCPG was coded in *Python*. The constant renewal of models and frameworks in NLP and deep learning in general makes it difficult to reimplement all architectures from scratch to test different scoring strategies. It was therefore decided to unify the different implementations available to build a flexible score module that could be quickly adapted to new models.

The idea is simple, we create a *Python* object for each sub-scorer in our scoring function. Each scorer is called independently on the whole batch. Finally the sub-scores are combined in a *scoring function*. In our module, we implemented scorers for the BERTscore, GPT-2, LANGUAGE-TOOL and the levenshtein distance that were used in our final scoring function. All those classes used already available implemented models.

We also added other scorers like :

— T2TScorer : a scorer that interfaces the Tensor2Tensor [Vaswani, S. Bengio, et al., 2018] library that implements the TRANSFORMERand SOTA neural networks architectures in TENSORFLOW

— FastTextScorer : a sorer that computes the cosine similarity between FastText embeddings [Bojanowski et al., 2017] of the two sentences. It relies on the official FastText repository. [1]

— BLEUScorer, ROUGEScorer, METEORScorer that leverage the *nlg-eval Python* API to use BLEU, ROUGE and METEOR metrics as the score for MCPG

Each one of these scorer modules generates a batch of score from a batch of pairs of paraphrases. Then, the resulting list of scores is feeded to a scoring function that computes the final batch of score.

To easily choose the scorers and scoring functions, we use a simple configuration files system using the YAML language.

1. https://github.com/facebookresearch/fastText

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : La Génération de Texte Basée Recherche et Supervisée

Mot clés : encodeur-decodeur, triplets RDF, paraphrase, recherche dans un arbre, distillation

Résumé : Les modèles supervisés **encodeurs-decodeurs** nécessitent de grands datasets alignées pour être entraînés. Les données nécessaires ne sont pas encore disponibles pour plusieurs tâches telles que la **verbalisation de triplets RDF** ou la **génération de paraphrases**. D'abord, nous avons exploré la tâche de verbalisation de triplets RDF. Nous avons entraîné des modèles **TRANSFORMER** sur une nouvelle version des données **WebNLG** et avons étudié plusieurs stratégies de pré-entraînement pour surmonter la petite taille du corpus. Ensuite, nous avons étudié la tâche de génération de paraphrases. Nous avons entraîné des modèles TRANSFORMER sur des corpus alignés afin de les comparer directement avec les modèles de la littérature. Une contribution importante de la thèse a été de proposer un **cadre expérimental uniforme** pour comparer les modèles encodeurs-décodeurs pour la génération de paraphrases. Nous avons également suivi la voie des méthodes alternatives basées recherche pour générer des paraphrases. Pour ce faire, nous avons transformer la tâche de génération de paraphrases en un problème de **recherche dans un arbre**. Nous avons ensuite développé deux stratégies de recherche : **MCPG** et **PTS** et un module de score des paraphrases qui exploite le **BERTscore**, **GPT-2** et la **distance de Levenshtein**. Enfin, nous avons mené des expériences de **distillation** avec le modèle TRANSFORMER.

Title: Search-Based and Supervised Text Generation

Keywords: encoder-decoder, RDF verbalization, paraphrase , TRANSFORMER, tree-search, distillation

Abstract: In this thesis, we studied the topic of *Search-Based and Supervised Text Generation*. Supervised **encoder-decoder** models require huge aligned dataset to be trained. The necessary data is not yet available for several tasks such as **RDF triples verbalization** or **paraphrase generation**. First, we explored the data-to-text task of RDF verbalization. We trained supervised **TRANSFORMER models** on a newly released version of the **WebNLG** dataset and studied in depth several pre-training strategies to overcome the small size of the aligned corpus. Then, we studied the paraphrase generation task. We have trained TRANSFORMER models on aligned corpora to directly compare with the literature model. An important contribution of the thesis was to propose a **uniform experimental framework** for comparing encoder-decoder models for paraphrase generation. We also followed the path of search-based alternative strategies to generate paraphrases. The main motivation was to provide a better control of the generated paraphrase. To do so, we casted the paraphrase generation tasks as a **tree-search** problem. We then, developped two search strategies **MCPG** and **PTS** and a paraphrase scoring module that leverages the **BERTscore**, **GPT-2** and the **Levenshtein distance**. Finally, we conducted experiments of data **distillation** for the TRANSFORMER model.