Traduction automatique neuronale avec JoeyNMT

Ahmed Belkhir

Résumé

Dans ce rapport, on présente le travail qu'on a fait dans le projet de traduction automatique neuronale de l'Anglais vers le Français basée sur le framework JoeyNMT. L Ce rapport présente l'état de l'art dans la traduction automatique avant d'expliquer l'environnement et le système qu'on a développé. On a obtenu un Bleu score de 21.95% avec notre système, ce qui est comparable aux systèmes avancés.

Mots clés: Traduction automatique, apprentissage profond, Joey NMT

1 Introduction

La traduction automatique est un processus par lequel un système traduit un texte d'une langue source à une autre langue cible sans intervention humaine. Ces systèmes là sont apparus dans les années 60s avec les traductions automatiques basées sur les règles. Puis, les traductions basées sur les exemples et les traductions automatiques statistiques ont été développées. Ces dernières années cependant, d'autres approches "neuronales" sont apparues et ont prouvé être encore plus efficaces surtout si l'on dispose d'un tas de données pour "entraîner" les modèles. Ce projet se base sur cette dernière approche pour développer un système de traduction automatique neuronal basé sur le framework Joey NMT dans le cadre du cours Traitement Automatique du langage Naturel (INF7546) enseigné à l'UQAM par Prof. Fatiha Sadat.

Dans la première partie de ce rapport, on va présenter les données qu'on a utilisées ainsi que la méthodologie suivie. Ensuite, on va présenter les résultats obtenus et les comparer aux autres systèmes développés pour la même base de données.

2 État de l'art

La première proposition de traduction automatique par ordinateur a été présentée en 1949 par Warren Weaver (Weaver, 1952). Les chercheurs ont essayé de développer différents systèmes de traduction automatiques comme SYSTRAN et Logos etc. et en 1990, Brown et. al ont proposé l'utilisation de méthodes statistiques dans les traductions automatiques (Brown et al., 1990). 3 années plus tard, ils ont décrit cinq modèles statistiques pour le processus de traduction et ont donné des algorithmes pour estimer les paramètres de ces modèles à partir d'un ensemble de paires de phrases bilingues. (Brown et al., 1993) Ces modèles ont ensuite été considérés comme les modèles d'alignement d'IBM. Récemment, et en 2013, une nouvelle structure d'encodeur-décodeur de bout en bout pour la traduction automatique a été proposée par Kalchbrenner Blunsom (Kalchbrenner and Blunsom, 2013). Une anée plus tard, Sutskever et al. ont proposé l'utilisation de réseaux neuronaux profonds dans l'apprentissage de séquence à séquence pour les traductions automatiques (Sutskever et al., 2014). En 2017, Vaswani et al, ayant vu les avanteges du mécanisme d'attention, ont proposé ce modèle pour la traduction automatique neuronale (Vaswani et al., 2017). Le transformateur est le même que le modèle séquence-à-séquence avec une paire encodeur-décodeur. Le codeur encode la séquence d'entrée et le décodeur génère la séquence de sortie. Cependant, ils ont trouvé un moyen d'utiliser le mécanisme d'attention au sein de l'encodeur et ont trouvé des résultats intéressants. Cependant, le développent de tels modèles n'est pas très facile, surtout pour les débutants. C'est pour cela que Joey NMT (Kreutzer et al., 2019) a été développé pour permettre aux débutants de développer un sytème de traduction automatique avec relativement peu d'effort. Plus de détails dans une section prochaine.

3 Données et ressources utilisées

3.1 Collection de données

Pour développer le système de traduction automatique, et comme on a choisi de faire une approche neuronale qui nécéssite beaucoup de données pour donner de bons résultats, on s'est basé sur la col-

lection de données Europarl V7 (Koehn and Monz, 2005) qui est un corpus parallèle extrait des actes du Parlement européen. Il comprend des versions dans 21 langues européennes. Dans notre travail, on a considéré la paire de langues Anglais-Français pour traduire de l'Anglais vers le Français. Le corpus compressé considéré est d'une taille d'environ 200Mo pour la paire de langues choisie et contient:

- 2 Millions de lignes de texte pour l'entraînement
- 3000 lignes pour le développement
- 3003 lignes pour le test

A titre d'exemple, voici une phrase en langue source (Anglais) du corpus d'entraînement:

"I declare resumed the session of the European Parliament adjourned on Friday 17 December *1999*, and I would like once again to wish you a happy new year."

La traduction correspondante en Français selon le même corpus:

"Je déclare reprise la session du Parlement européen qui avait été interrompue le vendredi 17 décembre dernier et je vous renouvelle tous mes *vux*."

Dans cet exemple, on peut remarquer qu'il y a deux fautes de traduction: l'année 1999 n'a pas été mentionnée dans la traduction en français, et le mot "veux" contient une faute d'orthographe. Cela peut nuire à la qualité du sytème de traduction qui se base sur des traductions non exactes.

3.2 Framework: Joey NMT

Joey NMT (Kreutzer et al., 2019), comme on a mentionné dans la section de l'état de l'art, est une boîte à outils de traduction automatique neuronale minimaliste basée sur PyTorch et spécialement conçue pour les débutants. Joey NMT fournit de nombreuses fonctionnalités interessantes de la traduction neuronale dans une base de code petite et simple, de sorte que les novices peuvent facilement et rapidement apprendre à l'utiliser et l'adapter à leurs besoins. Malgré sa simplicité, Joey NMT prend en charge les architectures d'apprentissage profond désormais classiques (RNN, transformateurs), la recherche rapide de faisceaux, la liaison de poids, etc., et obtient des performances comparables à celles des modèles les plus complexes. Les auteurs présument que leur outil peut atteindre 80% de la qualité de traduction avec 20% du code usuel. Le framework propose une multitude de paramètres pour jouer avec et voir leur impact sur la performance de la traduction.

3.3 Matériel utilisé

Afin d'entraîner le sytème de traduction automatique, on avait besoin d'avoir des machines puissantes avec des processeurs graphique (GPU) rapides et dontés d'une large mémoire vive (RAM). Pour cela on a opté pour Google Cola Pro qui ne coûtait que 16$ le mois et nous donne accès à un GPU Tesla P100 dotée de 16Go de RAM avec mémoire vive du système jusqu'à 25Go.

Cependant, le temps d'execution est limité à un nombre aléatoire d'heures. Ceci veut dire qu'on ne peut pas laisser le modèle qu'on développe s'entraîner pour une durée indéterminée sinon le GPU sera déconnecté et on aurait perdu le travail.

4 Méthodologie

Comme on a déjà mentionné précedemment, on s'est basé sur la plateforme Joey NMT pour développer le système de traduction automatique. On a commencé par installer les librairies nécéssaires avec les commandes linux suivantes:

```
pip install joeynmt
pip install torch==1.9.0+cu102
```

On a installé la version 1.9.0 de torch pour éviter les conflits de versions car Joey NMT a été développé pour cette version en particulier.

4.1 Tokenisation

On a utilisé la bibliothèque subword_nmt pour diviser les mots en sous-mots (BPE) en fonction de leur fréquence dans le corpus d'apprentissage.

Le principal avantage d'un tokenizer de sous-mots est qu'il interpole entre la tokenisation basée sur les mots et celle basée sur les caractères. Les mots courants obtiennent un emplacement dans le vocabulaire, mais le tokenizer peut se rabattre sur des morceaux de mots et des caractères individuels pour les mots inconnus.

Par d'exemple, la phrase "A Republican strategy to counter the re-election of Obama" devient en rajoutant "@@" entre les tokens: "A Re@@ publi@@ can strategy to counter the re-@@ election of Ob@@ ama" après tokenisation. Il est à noter que le découpage dépend de la taille du vocabulaire choisi. On a commencé par un vocabulaire de 4000 mots, puis on l'a augmenté à 20000 mots.

La tokenisation se fait bien sûr sur les deux langues du corpus parallèle.

4.2 Configuration du modèle

On a créé un fichier de configuration qui décrit le modèle qu'on veut créer. Joey NMT lit les hyperparamètres du modèle et d'entraînement à partir de ce fichier de configuration et construit le modèle. Les paramètres sont de 4 types:

4.2.1 Paramètres de données

Ce sont les paramètres qui spécifient les langues source et destination, les chemins des données d'entraînement, de développement et de test ainsi que la longeur maximale des phrases. Ce dernier paramètre spécifie la longuer de phrase en tokens après laquelle les tokens seront tout simplement ignorés. On spécifie aussi dans cette section d'autres paramètres comme les vocabulaires que le modèle va utiliser.

4.3 Paramètres de testing

Ce sont les paramètres qui spécifient les mesures qu'on va utiliser pour tester le modèle ainsi que quelques configurations comme beam_size qui spécifie combien de meilleures solutions partielles à évaluer.

4.4 Pramètres d'entraînement

C'est la plus grande partie du fichier de configuration. Elle comprend des paramètres importants et qui affectent l'entraînement tels que le taux d'apprentissage, le type de pertes (loss), la taille du batch, la fréquence de journalisation, le nombre d'itérations (appelés epochs)...

4.5 Paramètres du modèle

Dans cette partie, on détermine les spécifications du modèle d'apprentissage profond tels que le nombre de couches dans l'encoder et le décodeur, l'utilisation ou pas du mécanisme d'attention, le fast-forward size et la taille de dropout..

5 Évaluations et résultats

5.1 Mesure d'évaluation

Pour l'évaluation, on s'est basé sur le score BLEU (Bi-Lingual Evaluation Understudy) (Papineni et al., 2002). Il s'agit d'un moyen populaire et peu coûteux de mesurer automatiquement les performances d'un modèle de traduction automatique. En bref, le BLEU compare la traduction de la machine avec des traductions existantes générées par l'homme, appelées traductions de référence. Il calcule la précision - la fraction de tokens de la traduction candidate qui apparaissent, ou sont "couverts", par la traduction de référence.

5.2 Méthode de base

On a commencé d'abord avec les paramètres par défaut de Joey NMT, à savoir une taille du vocabulaire de 4000 mots et 10 epochs entre autres et max_sent_length égal à 30. Après un entraînemet avec les 2 millions de lignes et qui a duré 7 heures sur la configuration de Google Colab Pro, on a réussi à obtenir un BLEU de **17.69%** sur la collection de données de test et **16.34%** sur la collection de développement. Cependant, ce n'est sûrement le meilleur qu'on peut obtenir et on a essayé d'améliorer notre système en expérimentant avec les différents paramètres.

5.3 Les paramètres importants

Comme expliqué précedemment, le fichier de configuration de Joey NMT contient une trentaine de paramètes qui affectent le résultat final. On a expérimenté avec les hyperparamètres et on a constaté que certains paramètres affectent considérablement les résultats, tandis que d'autres n'ont pas d'effet visible.

Malheureusement, et faute de temps, on n'a pas de résultats finaux lors du changement des différents paramètres. On s'est contenté plutôt de changer un paramètre donné, de commencer l'entraînement et voir l'évolution du score BLEU au fil des itérations pendant quelques heures et de comparer cette évolution au journal (log) des systèmes précedents. Si le score évolue plus rapidement on laisse le modèle s'entraîner plus longtemps et si le score évolue lentement on arrête et on change les paramètre. Ceci nous a permis de conclure que les paramètres les plus importants sont:

- La taille des données d'apprentissage

Bien évidemment, plus on utlilise de données pour apprendre, plus le modèle construit sera meilleur. En contre partie, l'apprentissage sera lent.

- La taille du vocabulaire

Le paramètre bpe_size expliqué précedemment définit la taille du vocabulaire et a un effet considérable sur la performance du système de traduction automatique. On a essayé plusieurs

valeurs et on a conclu que l'incrémenter de 4000 à 20000 a donné une bonne performance. Cependant, en l'incrémentant d'avantage jusqu'à 50000 n'améliore pas les performances et ralentit l'entrainement.

- La taille maximale de la phrase

Incrémenter la valeur du paramètre max_sent_length a donné une amélioration considérable dans le score BLEU du système construit. On a choisi de fixer sa valeur à 100 au lieu de 30 par défaut.

- Architecture du modèle

On a doublé le nombre de couches, le nombre de heads et la taille de l'embedding dans les encodeurs et décodeurs et cela nous a amélioré le système final considérablement. En effet, cela rend le système plus complexe pour qu'il capte plus d'informations complexes dans la traduction.

5.4 Notre système de traduction automatique

Après avoir testé les différents paramètres, on a réussi à améliorer le système TA de base et on a atteint un score BLEU de **21.95%** sur les données de test fournis par WMT et un score BLEU de **19.80%** sur la collection de développpement. La configuration gagnante a été entaînée sur l'intégralité de la base de données Europarlv7, soit 2 millions de ligne. La taille du vocabulaire a été fixée à 20000 mots, la taille maximale des phrases à 100 tokens par phrases, et on a doublé le nombre de couches et le nombre de heads par rapport à la configuration par défaut.

Les courbes de pertes sur la collection d'entraînement et sur les données de développements sont présentés respectivement dans les figures 1 et 2. On peut voir clairement que plus le nombre d'itérations (pas le nombre d'ecpochs) est grand, plus les pertes sont réduites sur l'un ou l'autre courbe. On peut remarquer aussi qu'on atteint un plateau à la fin et que l'amélioration marginale est de plus en plus faible, mais non nulle.

Dans la figure suivante (figure 3), on peut voir que le score de validation augmente rapidement avec les premières itérations, puis ralentit peu à peu jusqu'à atteindre un plateau lui aussi.

Pour atteindre le score de presque 22%, l'entraînement a duré 9 heures sur le GPU Nvidia Tesla P100, et on a arrêté l'entrainement de perte que Google Colab se déconnecte comme il l'a fait

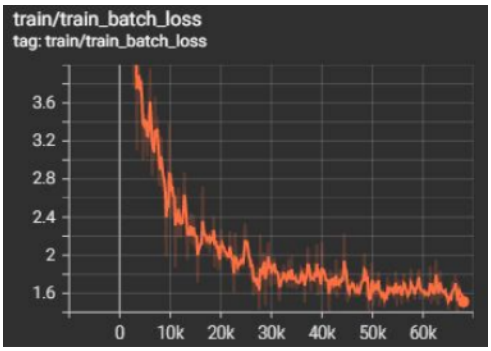

Figure 1: L'évolution des pertes dans la collection d'entraînement

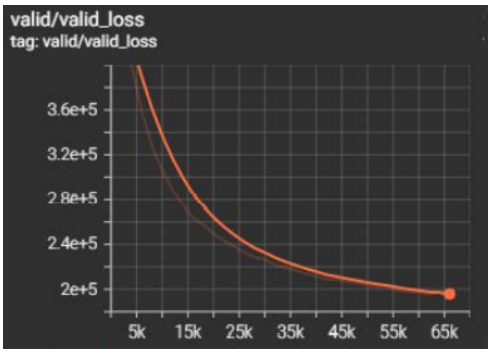

Figure 2: L'évolution des pertes dans la collection de développement

tant de fois. Cependant, le résultat était pas mal surtout comparé aux systèmes développés par les professionnels.

5.5 Comparaison des résultats

Le meilleur sytème développé dans la compétition de WMT14 pour la traduction de l'Anglais vers le Français ont atteint 32.7% pendant l'année 2014, ce qui n'est pas trop loin de notre système développé avec des limites plus importants en temps d'entraînement et en puissance de calcul.

On peut voir cela dans le tableau 1 issu de l'article (Bojar et al., 2014) les résultats des différents systèmes développées pour la compétition de 2014 sur la même tâche que la nôtre.

Plus récemment, d'autres systèmes de traduction automatique plus sophistiqués ont été développés et ont atteint un BLEU score de 46.4%. On peut voir les performances des systèmes les plus avancés dans la figure 2. Il faut se rappeler toujours que les meilleurs sytèmes sont développées avec une configuration de 128 GPU et dans des cas avec des corpus de 137 millions de paires de phrases (Edunov et al., 2018), ce qui est loin de ce qu'on dispose pour ce projet.

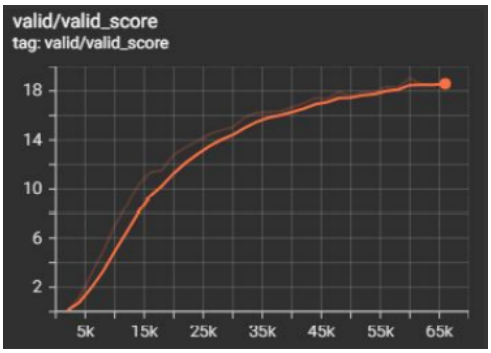

Figure 3: L'évolution des pertes dans la collection de développement

Transformer+BT (ADMIN init)	46.4	44.4	✓	Very Deep Transformers for Neural Machine Translation
Noisy back-translation	45.6	43.8	✓	Understanding Back-Translation at Scale
mRASP+Fine-Tune	44.3	41.7	✓	Pre-training Multilingual Neural Machine Translation by Leveraging Alignment Information
Transformer + R-Drop	43.95		×	R-Drop: Regularized Dropout for Neural Networks
Transformer (ADMIN init)	43.8	41.8	×	Very Deep Transformers for Neural Machine Translation
Admin	43.8		×	Understanding the Difficulty of Training Transformers
BERT-fused NMT	43.78		×	Incorporating BERT into Neural Machine Translation

Figure 4: L'évolution du score BLEU au cours des itérations (https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-french)

English–French

#	score	range	system
1	0.327	1	ONLINE-B
2	0.232	2-4	UEDIN-PHRASE
	0.194	2-5	KIT
	0.185	2-5	MATRAN
	0.142	4-6	MATRAN-RULES
	0.120	4-6	ONLINE-A
3	0.003	7-9	UU-DOCENT
	-0.019	7-10	PROMT-HYBRID
	-0.033	7-10	UA
	-0.069	8-10	PROMT-RULE
4	-0.215	11	RBMT1
5	-0.328	12	RBMT4
6	-0.540	13	ONLINE-C

Table 1: Résultats officiels de la tâche de traduction WMT14 (Bojar et al., 2014)

Conclusion et perspectives

Ce projet a été une bonne occasion pour apprendre beaucoup de choses. D'abord, on s'est familiarisé avec la manipulations des corpus parallèles et avec l'entraînement des modèles d'apprentissage profonds. Il a été important d'essayer de développer un système de traduction automatique avec des performances moyennes en un temps très raisonnable. Pour améliorer encore ce système, on peut laisser l'entraînement se faire pendant encore un peu plus de temps et on pourrait avoir un score un peu plus élevé. Cependant, pour avoir un système TA bien avancé, il est inévitable de développer son propre système dès le début sans librairies comme Joey NMT pour avoir plus de possiblités de personnalisation.

On veut conclure avec la citation du fondateur de l'internet Tim Berners-Lee et sa traduction avec le système TA qu'on a dévelppé dans ce projet:

> –"I don't know whether machine translation will eventually get good enough to allow us to browse people's websites in different languages so you can see how they live in different countries."
> –"Je ne sais pas si la traduction de la machine nous permettra de briser les sites web des citoyens dans différentes langues. Vous pouvez donc voir comment ils vivent dans différents pays."
>
> *Tim Berners-Lee*

References

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Philipp Koehn and Christof Monz. 2005. Shared task: Statistical machine translation between european lan-

guages. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 119–124.

Julia Kreutzer, Jasmijn Bastings, and Stefan Riezler. 2019. Joey nmt: A minimalist nmt toolkit for novices. *arXiv preprint arXiv:1907.12484*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Warren Weaver. 1952. Translation. In *Proceedings of the Conference on Mechanical Translation*.