Projet M1 TAL - Rapport du mi-semestre

Réseau de neurones pour le résumé automatique de phrases basé sur la suppression

Yiming Liang & Fang Zhao
Sous la direction de Benoît Crabbé

11 avril 2020

1 Introduction

La compression automatique d'une phrase consiste à générer une phrase plus courte en gardant l'essentiel de la phrase originale. Servant potentiellement à générer des résumés de texte, celle-ci permet aux lecteurs de mieux traiter de l'information d'une longue phrase en très peu de temps, une tâche qui est fortement requise dans un contexte où une masse d'informations ne cesse de pénétrer la vie en raison du développement et de la démocratisation de l'Internet.

En effet, le résumé automatique de phrases a agité beaucoup de chercheurs dans le domaine du traitement automatique des langues ces vingt dernières années, où divers systèmes ont été avancés pour traiter ce problème. Une des manières les plus utilisées consiste à condenser une phrase en supprimant des mots superflus. Ayant été fortement dépendante de l'arbre syntaxique (Jing, 2000 ; Knight & Marcu, 2000), cette technique semble capable de produire des résumés de qualité sans avoir besoin d'informations linguistiques grâce à l'utilisation de réseaux de neurones (par exemple *Long short-term memory (LSTM)* (Filippova, Alfonseca, Colmenares, Kaiser, & Vinyals, 2015) et plus généralement *Recurrent neural network (RNN)* avec mécanisme d'attention (Rush, Chopra, & Weston, 2015 ; Chopra, Auli, & Rush, 2016). Cependant, des études plus récentes montrent que l'introduction des connaissances linguistiques dans le réseau de neurones LSTM, telles que les arbres

syntaxiques, sont très utiles pour réduire les données nécessaires à l'apprentissage et améliorer la performance du système de compression (Wang et al., 2017). D'autres chercheurs montrent également que la psycholinguistique, comme la technique de eye-tracking, aide dans la simplification de phrase (Klerke, Goldberg, & Søgaard, 2016).

Ainsi se pose la question : en quelle mesure les informations linguistiques contribuent à la tâche de compression de phrase réalisée à l'aide d'un réseau de neurones ? Dans ce projet, nous cherchons à répondre à cette question à travers des expériences sur un modèle de compression basés sur la suppression. Partant du celui proposé par Filippova et al. (2015) qui ne dépendent que de l'information séquentielle de phrase, nous nous proposons d'améliorer ce modèle surtout à l'aide de l'information syntaxique, comme proposé dans Wang et al. (2017), afin d'augmenter la performance de compression automatique. De plus, nous souhaiterons également trouver des caractéristiques des mots ayant tendance à être enlevés par le système de compression, afin de voir si ces résultats correspondent aux besoins du résumé de l'homme, et peuvent mettre en lumière le mécanisme de résumé utilisé par l'homme.

Nous envisageons d'utiliser *Google News Dataset* (Filippova & Altun, 2013), une base de données comprenant 200 000 paires de phrases (phrase d'origine et phrase compressés) comme données d'apprentissage et 10 000 paires de phrases comme données de test. Concrètement, les tâches que nous souhaiterons réaliser dans ce projet sont présentées comme ci-dessous :

1. implémentation du modèle proposé par Filippova et al. (2015) sans usage d'information linguistique comme un modèle de base. Il s'agit d'une tâche de l'apprentissage supervisé à travers le réseau de neurones *Long Short-Term Memory (LSTM)* dans le cadre de la Seq-to-Seq. Nous comptons également d'employer un LSTM bidirectionnel (biLSTM) pour tester sa performance.

2. introduction dans ce modèle de base des vecteurs apportant des informations linguistiques de mots ou de la phrase, telles que les catégories syntaxiques et les arbres en dépendances, afin de comparer la performance de ce modèle avec celle du modèle original (i.e. celui de Filippova et al. (2015)).

3. montage des résultats issus des expériences surtout avec le modèle de compression automatique nourri par des informations linguistiques. Nous envisageons d'étudier les points communs parmi les mots ou les parties ayant tendance à être supprimés

lors du résumé automatique de phrase, en nous interrogeant en particulier sur les questions suivantes : est-ce que ceux-ci sont conformes à la théorie de l'information ? Les noeuds situés moins proche de la racine de l'arbre en dépendances sont-ils plus probables d'être enlevés ?

4. si on a le temps, construction d'un nouveau modèle qui lit la phrase en descendant l'arbre en dépendances, au lieu de lire la phrase de manière séquentielle.

5. si on a le temps, conception d'une nouvelle métrique d'évaluation qui répond mieux aux besoins du résumé de phrase de l'humain comme suggéré dans Peyrard (2019), à la lumière des nouveaux avancements dans le domaine de la théorie de l'information, voire une fonction de perte à partir de ceux-ci.

2 Étude pilote

Afin d'évaluer le temps dont nous aurons besoin pour réaliser les tâches mentionnées ci-dessous et de prévenir des difficultés que nous pourrions rencontrer lors de l'implémentation d'un modèle LSTM dans le cadre de la Seq2seq, nous avons effectué une étude pilote où nous avons implémenté le modèle de base, i.e. celui de Filippova et al. (2015) sans aucunes informations linguistiques sauf la tokénisation, avec *PyTorch*. Nous avons aussi, à la suite de l'implémentation de la recherche en faisceaux (beam search), testé la performance de ce modèle sur l'ensemble de l'apprentissage et l'ensemble de test. Avant de détailler cette étude pilote, il convient de présenter brièvement le modèle LSTM et la notion de Seq2seq.

L'architecture de LSTM (Long Short-Term Memory) [1] (Hochreiter & Schmidhuber, 1997) est un type particulier du modèle RNN (Recurrent Neural Network). Un RNN est un réseau neuronal récursive qui permet de mémoriser un contexte plus large qu'un simple réseau de neurones tel que MLP (Multilayer Perceptron), mais il ne fonctionne pas très bien lorsque la relation de dépendance entre deux mots est très longue. Un grand avantage d'un LSTM par rapport à un simple RNN est qu'il peut stocker des longues dépendances dans la phrase en entrée, grâce à un mécanisme de portes (*gates*) qui contrôle la mise à jour de la mémoire à chaque moment qu'un nouvel élément de l'input est entré dans le réseau. Plus précisément, ces différentes portes vont décider quelle partie des éléments déjà stockés dans

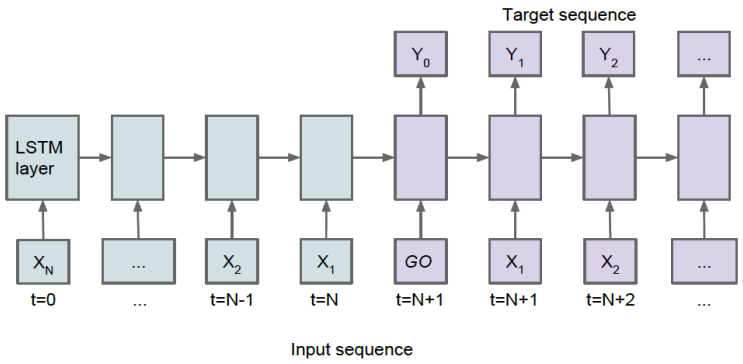
1. Il existe plusieurs variantes de LSTM.

la mémoire sera oubliée et quelle partie de l'input sera stockée dans la mémoire. Grâce à cet avantage, le LSTM est très convenable à beaucoup de tâches du traitement automatique des langues comme la traduction automatique, la prédiction de texte et le résumé automatique comme ce qui est présenté dans ce projet.

Seq2seq (*sequence to sequence*) (Sutskever, Vinyals, & Le, 2014) est un paradigme à la recherche des paramètres qui maximisent la probabilité de la séquence de l'output étant donné la séquence de l'input. Plus précisément, étant donné une séquence d'entrée X et une séquence de sortie Y, il faut que les paramètres du modèle rendent maximale la somme des $P(Y|X)$ pour tous les exemples d'apprentissage :

$$\theta^* = \arg\max_{\theta} \Sigma_{X,Y} log\ p(Y|X;\theta)$$

Dans notre étude pilote, le vecteur de chaque mot de l'input (i.e. la phrase originale) contient 260 dimensions, dont 256 constituent la représentation vectorielle du mot courant et 4 la représentation one-hot de l'étiquette du mot précédent (i.e. suppression, préservation, frontière de phrase ou mot de remplissage dans le batch 'pad', contre 3 dans le modèle original). La représentation vectorielle du mot courant est issu du modèle *GloVe* (Pennington, Socher, & Manning, 2014) pré-entraîné (840B 300d) au lieu du modèle *Skipgram*. Comme détaillé dan Filippova et al. (2015), la séquence de l'input a été passée deux fois dans le modèle. Lors du premier passe, les mots de la phrase d'entrée sont introduits dans l'encodeur en ordre inverse, qui est composé de trois couches de LSTM entrelacées avec des couches "dropout" qui abandonnent aléatoirement une partie des informations entre deux couches LSTM, afin d'éviter le sur-apprentissage (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Au bout de cette étape, l'encodeur est censé de renvoyer une représentation compacte de la phrase en entrée. Cette représentation entre ensuite dans le décodeur qui, lors du deuxième passe de la phrase en entrée en ordre normal, prédit si un mot de l'input sera supprimé ou sauvegardé dans la sortie (i.e. la phrase compressée) par la fonction *SoftMax* et la recherche en faisceaux.



L'expérience s'est effectuée sur un sous-ensemble de Google News Dataset. Nous avons utilisé un échantillon de 2100 paires de phrases originales et compressées, soit 1% des exemples de la base de données. La taille des ensembles d'apprentissage, de validation et de test est respectivement de 1800, 200 et 100 exemples. Le modèle est entraîné pendant 20 époques (nous avons choisi l'optimiseur Adam au lieu de SGD, qui a pour avantage d'accélérer énormément le processus d'apprentissage) sur 1800 exemples dans Google Colaboratory.

La perte *NLLLoss* (*Negative Log Likelyhood Loss*) sur l'ensemble d'apprentissage est de 0.1 avec beam = 1 et beam = 3, ce qui montre que notre modèle a réussi à faire des compressions phrastiques sur ces 1800 exemples au bout de 20 époques. La perte sur l'ensemble de test est de 0.389 avec beam search (beam=3) et 0.482 sans celle-ci (équivalant beam=1). Nous pouvons constater que, bien que la valeur de beam n'influence pas l'évaluation sur l'ensemble d'apprentissage quand l'échantillon est petit, le modèle appris avec une valeur de beam plus élevée a une meilleure performance dans le test. Pour le moment, nous n'avons pas encore répété l'expérience qu'a fait Filippova et al. (2015) avec les données en entier (2 millions d'exemples, dont 20 000 publiés en ligne) pour des raisons de pouvoir de calcul.

3 Limites et difficultés

Au cours de notre étude pilote, nous avons notamment rencontré les difficultés suivantes :

1. Pouvoir de calcul. Bien qu'on ait employé Google Colaboratory pour une implémentation sur GPU, le pouvoir de calcul paraît toujours faible par rapport à ce que nous

essayons de faire dans ce projet : sur notre modèle LSTM, avec les hyperparamètres de défaut, dont notamment beam=1 pour la partie décodage avec beam search, l'entraînement d'une époque avec la totalité des données de Google News Dataset prendrait 1 heure, voire plus. Les pistes de remédiation qu'on a fixées pour le moment consistent à chercher une implémentation de la recherche en faisceaux plus efficace qui remplacera notre implémentation vanille, ainsi que d'essayer de faire une implémentation sur TPU, qui serait "15 - 30 fois plus rapide que le GPU".

2. Beam search avec batch size > 1. Les pseudo-codes décrivant l'algorithme de la recherche en faisceaux dans la phase décodage donnés dans Filippova et al. (2015) ne précisent pas les mesures à prendre lorsque la prédiction s'effectue sur des batches de données, i.e. lorsqu'il existe plusieurs phrases à traiter comme une seule entrée. Alors que la recherche en faisceaux ne garde que les N suites de prédictions les plus probables, il est problématique de décider ce que cela vaut pour plusieurs phrases. Pour une solution temporaire, nous avons décidé de prendre la moyenne des probabilités des phrases. Nous allons continuer à explorer de meilleures solutions.

3. Étiquetage du statut de suppression des mots dans la phrase originale par rapport à la phrase compressée par suppression. Nous n'avons pas trouvé de fonctions "prêtes à employer" pour ce but. Pour notre étude pilote, nous avons choisi d'implémenter une version naïve sans traiter les exceptions telles qu'au cas où a lieu le désaccord de la tokenisation entre les phrases source et compressée. Nous allons compléter notre modèle de base avec une fonction d'étiquetage mieux définie.

4 Ce qui reste à faire

1. Changer le LSTM à biLSTM pour voir si la performance du modèle de compression s'améliore.

2. Introduire des informations linguistiques, surtout syntaxiques, dans le modèle de base pour l'améliorer.

3. Évaluer et comparer ces différents modèles.

4. Essayer de trouver les caractéristiques des mots qui tendent à être supprimés lors du résumé.

Références

Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics : Human language technologies* (pp. 93–98).

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, Ł., & Vinyals, O. (2015). Sentence compression by deletion with lstms. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 360–368).

Filippova, K., & Altun, Y. (2013). Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1481–1491).

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

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Sixth applied natural language processing conference* (pp. 310–315).

Klerke, S., Goldberg, Y., & Søgaard, A. (2016). Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv :1604.03357*.

Knight, K., & Marcu, D. (2000). Statistics-based summarization-step one : Sentence compression. *AAAI/IAAI*, *2000*, 703–710.

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)* (pp. 1532–1543).

Peyrard, M. (2019). A simple theoretical model of importance for summarization. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 1059–1073).

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv :1509.00685*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Wang, L., Jiang, J., Chieu, H. L., Ong, C. H., Song, D., & Liao, L. (2017). Can syntax help ? improving an lstm-based sentence compression model for new domains.