Note technique sur le traitement de langage naturel (TLN)
Ministère de la Transition Ecologique
CGDD - Ecolab

Théo Roudil-Valentin*

Les modèles de *Neural Machine Translation* ayant mené au modèle BERT, et à son efficacité

Les modèles de *Word Embedding* (vectorisation de mots) permettent de mettre des mots sous forme de vecteurs. C'est le cas par exemple du modèle canonique *Word2Vec* de Mikolov et al. [2013b,a] ou encore de *GloVe* de Pennington et al. [2014]. Ces modèles permettent donc de projeter des mots dans des espaces de dimension $d \in \mathbb{N}^{*+}$. Par exemple, pour le modèle de Mikolov, il est habituel de choisir $d = 300$. Ce qui revient à dire que chaque mot est représenté par un vecteur de 300 chiffres. C'est modèles sont intéressants pour, notamment, deux raisons. Ils s'éloignent des représentations standards type *Bag-of-Words* (sac de mots), car ils analysent les mots dans leur contexte et non de manière isolée. Ensuite, grâce à un d élevé, ils permettent de garder beaucoup d'information pour chaque mot, et ainsi avoir une richesse sémantique sous formes de vecteurs. Mais ces modèles ont aussi des inconvénients : ils ne peuvent pas faire de la représentation vectorielle de **phrases** (ou *Sentence embedding*).
En effet, pour cela, il convient d'élargir le modèle pour qu'il prenne en compte l'ensemble de la phrase, des modèles spécialisés dans la vectorisation de phrase ont été proposés [Le and Mikolov, 2014], et cela pose de nouveaux problèmes. Un des problèmes majeur est *the curse of sentence length* ou la malédiction de la longueur de phrase. Plus la phase est grande, plus le modèle devra sacrifier d'information dans la projection de la phrase dans un espace de dimension donnée, comme observé par Cho et al. [2014]. Le modèle standard de ce type est le *RNN Encoder-Decoder* : un encodeur lit la séquence d'entrée $\mathbf{x} = (x_1, ..., x_{T_x})$ et la transforme en un vecteur c, de dimension fixe. L'approche la plus répandue est d'utiliser un RNN ($RecurrentNeuralNetwork$) :

$$h_t = f\left(x_t, h_{t-1}\right)$$

et

$$c = q\left(\{h_1, ..., h_{T_x}\}\right)$$

avec $h_t \in \mathbb{R}^n$, une couche/neurone caché à l'instant t, et c un vecteur généré par la séquence de ces neurones cachés. f et q des fonctions non-linéaires. Le décodeur est entraîné à prédire le prochain mot $y_{t'}$ via le vecteur de contexte c ainsi que les mots prédits précédemment. Le décodeur tente de définir une probabilité pour une traduction $\mathbf{y}$, donc une séquence $(y_1, ..., y_{T_y})$,

*theo.roudil-valentin@ensae.fr

en décomposant la probabilité jointe de manière ordonnée :

$$p(\mathbf{y}) = \prod_{t=1}^{T} p\left(y_t | \{y_1, ..., y_{t-1}\}, c\right) \tag{1}$$

De même en utilisant à nouveau un RNN, chaque probabilité est définie comme :

$$p\left(y_t | \{y_1, ..., y_{t-1}\}, c\right) = g\left(y_{t-1}, s_t, c\right) \tag{2}$$

avec g une fonction non-linéaire, potentiellement multi-couche, qui donne donc la probabilité du mot y_t, et s_t la couche cachée du RNN (cette partie est largement reprise de l'article [Bahdanau et al., 2014], dont l'effort de clarté est apprécié).
Pour dépasser ce problème, Bahdanau et al. [2014] proposent une nouvelle architecture de modèles permettant de gérer les phrases longues. La première modification d'ampleur, est la modification du vecteur de contexte : celui-ci va dépendre du mot cible de sorti. Autrement dit, pour chaque y_i il y a un vecteur de contexte c_i associé. Chaque c_i dépend de la séquence de couche cachée (*annotations*) $(h_1, ..., h_{T_x})$, dont chaque élément contient des informations sur l'ensemble de la séquence $\mathbf{x}$ mais avec une concentration d'informations sur les mots proches du i-ème mot. Ainsi la probabilité est modifiée ainsi :

$$p(y_i | y_1, ..., y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_{i,c}\, i)$$

avec $s_i = f(s_{i-1}, y_{i-1}, c_i)$.
Le vecteur de contexte est calculé ainsi :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{3}$$

ainsi, les c_i peuvent être considérés comme des *annotations espérées* (*expected annotation*), avec chaque poids comme probabilité que le mot cible y_i soit ajusté (juste) pour la traduction du mot x_j. Ainsi, le vecteur contexte est bien l'annotation espérée, somme de chaque annotation pondérée par la probabilité d'alignement. Chaque poids est modélisé comme suit :

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

et représente l'importance de l'annotation j par rapport à la couche cachée précédente, s_{i-1}, dans la sélection de la nouvelle couche cachée s_i et donc du mot final y_i. C'est la modélisation du principe d'**attention**. Le décodeur, qui génère donc la séquence de sortie, décide, seul, des parties de la séquence d'entrée auxquelles il doit faire attention, via cette modélisation plus flexible. Ce qui permet cette attention, c'est la modélisation des e_{ij} :

$$e_{ij} = a\left(s_{i-1}, h_j\right)$$

avec a un modèle d'*alignement* de type *Feedforward Neural Network (FNN)* : qui va produire un score représentant à quel point les intrants autour de j et l'extrant en position i (en sortie, donc) sont cohérents. Ce n'est pas le mot y_i qui est utilisé, mais la couche cachée précédente s_{i-1}. Ce processus va donc permettre de produire, le poids correspondant α_{ij}, qui va lui-même influencer le vecteur de contexte c_i et donc la prochaine couche cachée s_i qui va elle-même produire le mot cible y_i. Il y a donc bien une idée de contexte, de manière bi-directionnelle, comme détaillée plus bas. De plus, c'est véritablement cette fonction a qui permet de relâcher l'obligation d'une

dimension fixe pour l'encodage de l'information. Le modèle d'alignement n'est plus une variable latente, comme dans l'approche précédente, mais bien quelque chose qui va être entraîné et optimisé avec le reste.
La deuxième importante modification est la **bi-directionnalité**. Dans le modèle précédent, le *RNN* lit la séquence d'entrée dans l'ordre, du x_1 jusqu'à x_{T_x}. Donc celui-ci ne prend pas en compte les mots qui suivent un mot considéré. La nouvelle approche au contraire utilise un réseau de neurones récurrents bi-directionnel (BiRNN, proposé par Schuster and Paliwal [1997]), qui permet justement de prendre en compte les mots qui suivent.
Ce type de réseau est constitué de deux réseaux, l'un *vers l'avant* l'autre *vers l'arrière*. La différence est donc le sens dans lequel le réseau va lire la séquence d'entrée. Le premier, $\overrightarrow{f}$, lit la séquence $\mathbf{x}$ de gauche à droite $(x_1, ..., x_{T_x})$; le deuxième, $\overleftarrow{f}$, la lit dans l'autre sens $(x_{T_x}, ..., x_1)$.Pour chaque intrant x_j de la séquence $\mathbf{x}$, on a donc deux neurones cachés : $\overleftarrow{h}_j$ et $\overrightarrow{h}_j$. Ces deux neurones cachés vont ensuite être pris ensemble pour créer une seule couche h_j :

$$h_j = \left[\overrightarrow{h}_j^T; \overleftarrow{h}_j^T\right]^T$$

La couche cachée finale contient donc les informations sur les mots passés et à venir : c'est le côté bi-directionnel de l'approche.
Les auteurs [Bahdanau et al., 2014] montrent que leur approche surpasse les précédentes approches, et notamment sur les phrases longues, qui était une des grandes difficultés à dépasser.

BERT : Bidirectional Encoder Representations from Transfomers, le modèle-frontière en NLP

Le modèle BERT canonique

Vaswani et al. [2017], Devlin et al. [2018]

Le modèle français : CamemBERT

Martin et al. [2019]

Utiliser les modèles BERT : *pre-training* et *fine-tuning*

Sun et al. [2019], Peters et al. [2019], Howard and Ruder [2018]
Nous on a besoin de faire :

— extractive text summarization

— text similarity

Références

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv :1409.0473, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation : Encoder-decoder approaches. arXiv preprint arXiv :1409.1259, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv :1810.04805, 2018.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. arXiv preprint arXiv :1801.06146, 2018.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In International conference on machine learning, pages 1188–1196. PMLR, 2014.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert : a tasty french language model. arXiv preprint arXiv :1911.03894, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv :1301.3781, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv :1310.4546, 2013b.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove : Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune ? adapting pretrained representations to diverse tasks. arXiv preprint arXiv :1903.05987, 2019.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11) :2673–2681, 1997.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification ? In China National Conference on Chinese Computational Linguistics, pages 194–206. Springer, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv :1706.03762, 2017.