Learn the Dependencies Between Content Words and Function Words

Junyang Lin
School of Electronics Engineering and Computer Science, Peking University
{linjunyang}@pku.edu.cn

Abstract

Based on a linguistic theory that content words in a sentence carry most of the semantic meaning of the whole sentence while function words mostly perform syntax-relevant functions, this article proposes an end-to-end model of neural network for both machine translation and abstractive summarization, which can self-adaptively learns the paradigms of dependencies between function words and content words. Therefore, with the learned dependencies, there is no need to feed all the inputs into the RNN encoder, which may produce high computation cost. Tests on the datasets, WMT-14 English-French and English-German for machine translation as well as Gigaword and LCSTS for abstractive summarization, show that our model outperforms the baseline models and achieves the state-of-the-art performance. Moreover, our model contains the advantage of computation efficiency over most of the end-to-end models for machine translation and abstractive summarization.

1 Introduction

The linguistic issue of content word and function word has been discussed for thousands of years, from Aristotle's period to today. There are a lot of other names for the two terms, such as lexical word and grammatical word (Lyons, 1968; Bybee et al., 1994), semantic word and grammatical word (Saeed, 1997) and so on. It is reasonable to categorize words into the two classes for the reason that many words deliver conceptual meaning, which are of a large amount, while some others remain relatively stable and function as tools to build syntactic-relevant structures. Talmy (2000) proposed that a fundamental design feature of language is that it contains two subsystems, which are lexical and grammatical systems. They have distinct semantic functions and they are indispensable and complementary. Following the commonly accepted idea in linguistics, human beings divide words into content word and function word. Content words are nouns, verbs or adjectives that represent semantic meaning, which are in an open class that the number of words can be increased. On the contrary, function words usually help to build syntactic structures for sentences depending on the contexts, which are in a closed class that the number of the words is stable.

Due to the significance of the differences between the two classes of words, this issue should be concerned in Natural Language Processing (NLP), especially in Natural Language Generation (NLG). As NLG aims at generating reasonable and readable texts, it should take both semantic meaning and syntactic structure into concern, which means researchers in this field need to pay attention to the features and functions of content words and function words. However, most NLP systems today did not process the two classes of words separately. Take machine translation and single-document abstractive summarization as examples. The relevant researches in Deep Learning mostly treat a sentence or a text as a whole and process them together with a sequence-to-sequence model (Sutskever et al., 2014a). Sequence-to-sequence model is an end-to-end encoder-decoder model (Cho et al., 2014a), which can encode a source text to a compressed feature vector which represents the meaning of the source text and then send it into the decoder to decode the desired output. This structure seems perfect for the two tasks, but it to some extent turns the process into a black box, making it hard for researchers to find out how the machine can understand the meaning of the source text and generate a correct and readable translation or a concise and reasonable summary. Moreover, the basic sequence-to-sequence model cannot figure out the syntactic structural information. Attention-based models can to some extent alleviate this problem (Bahdanau et al., 2014; Luong et al., 2015a), but better methods for the machine to understand the syntactic structures are still required.

Another prominent problem in this processing method is computation efficiency. Some previous studies show that there is no need for the encoder to process all the input source text with the purpose of finding the good compressed feature vector for the decoder to generate the desired output. A large amount of information in the source text may be useless for this purpose so that it can be better for the model to learn what information it needs to encode by itself (Yu et al., 2017; Johansen and Socher, 2017). These models can learn to skim text by ignoring certain words in the text or simply encode the text with the bag-of-word (BOW) encoder instead of Recurrent Neural Network (RNN). However, although the articles state that they can learn the rules self-adaptively, in fact they still highly depend on the choice of hyperparameter. Moreover, sequential skimming may not be the thinking patterns of human beings for reading texts. For a sentence, humans can figure out the syntactic structure and understand the meaning through the syntax-semantic interface with their own linguistic competence (Hackl, 2013).

This article proposes a method to tackle the problem by processing content words and function words in the source text separately. Following the ideas in dependency theory (Hays, 1964), there are various kinds of ideas of dependency in language. We formulate that since function words have close connections with syntax while content words are related to semantic meaning, dependencies between content words and function words can be formed and help the system to process texts. In the syntactic dependencies, it can be found that in many cases content words depend on function words in the tree structures. Therefore, with the hypothesis of this type of dependency, our model processes the content words with a BOW encoder and the function words with an RNN encoder plus a proposed attention mechanism to capture syntactic dependencies. This model can not only process the information in the source text with the consideration of both semantic meaning and syntactic structure, but also improves computational efficiency to up to x times owing to separate processing. Tests on both the tasks of machine translation and abstractive summarization all show that our model has significant advantages of performance and efficiency over the baseline models and outperforms the state-of-the-art techniques.

2 Proposed Model

Machine translation and abstractive summarization can both be treated as a sequence-to-sequence mapping task, which maps the source text and the target text. In machine translation, the model translates the text in the source language to that in the target language, while in abstractive summarization, the model generates the summary based on the input text. Therefore, our model is based on traditional sequence-to-sequence model (Sutskever et al., 2014a), which contains an encoder and a decoder. Moreover, the popular attention mechanism is also applied in order to capture the relevant source information in the process of decoding, and global attention is used in our model (Luong et al., 2015a). However, different from the previous work, our encoder processes the content words and the function words in the source text separately, and there are also corresponding changes in our attention mechanism.

For the separate encoding, our encoder contains two parts for the content word sequence and the function word sequence, still preserving the original sequential order. For the content word sequence, since our goal is to retrieve the semantic meaning, which has relatively weak relation to sequential order, there is no necessity to process it with an RNN encoder, which has relatively high computational costs and slow processing speed. Following the idea of BOW, we simply process the content word sequence with a Multilayer Perceptron (MLP) (Rosenblatt, 1960). For the function word sequence, as it has fundamental connections with the syntactic structure, which is sensitive to the sequential order, it is processed by an RNN encoder. We implement a bidirectional Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997). However, the input for this encoder is not simply the word embeddings of the function words. Instead, each function word is required to attend the content words, and generate a context vector as most attention mechanisms do. Then the embedding and the context vector are concatenated and fed into a MLP and our implemented bidirectional LSTM. As to the decoder, similar to most of the previous work, the decoder generates words sequentially, one at a time step, but in the generation it attends to the two sequences to calculate attentions and generate the context vectors.

2.1 Encoder

Before encoding, the source text is preprocessed by extracting function words and build the function word sequence of length n and the content word sequence of length m. The vocabulary of function words are retrieved from the NLTK stopword corpus (Bird and Loper, 2004). Words in the sequences are embedded in the same semantic space and the word embeddings are sent into the encoder. The two sequences are sent into separate parts of the encoder. For the content word sequence, the word embeddings are fed into a 2-layer MLP with the ReLU function (Glorot et al., 2011) as the activation function, which enables the words in the same source text to share the same parameters in the MLP. It generates an encoder output o_t at each time step t, which will be processed by the second part of the encoder in the next process.

For the function word sequence, we implement a 2-layer bidirectional LSTM for the processing. Similar to traditional RNN encoder, it receives the inputs sequentially and generates an encoder output as well as a hidden state and a cell state at each time step. The final states are sent into the decoder for the sequence generation. Bidirectional LSTM is an LSTM which processes the sequence in two directions and usually concatenates or element-wise add the encoder outputs from both directions at the same time step, and the LSTM is defined as below.

$$f_i = \sigma(\mathrm{W_f}[x_i, h_{i-1}] + b_f) \tag{1}$$
$$i_i = \sigma(\mathrm{W_i}[x_i, h_{i-1}] + b_i) \tag{2}$$
$$o_i = \sigma(\mathrm{W_o}[x_i, h_{i-1}] + b_o) \tag{3}$$
$$\tilde{\mathrm{C_i}} = \tanh(\mathrm{W_C}[x_i, h_{i-1}] + b_C) \tag{4}$$
$$\mathrm{C_i} = f_i \odot \mathrm{C_{i-1}} + i_i \odot \tilde{\mathrm{C_i}} \tag{5}$$
$$h_i = o_i \odot \tanh(\mathrm{C_i}) \tag{6}$$

Bidirectional LSTM generates two sequences of hidden states $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, ..., \overrightarrow{h_n}\}$ and $\overleftarrow{h} = \{\overleftarrow{h_1}, \overleftarrow{h_2}, \overleftarrow{h_3}, ..., \overleftarrow{h_n}\}$, and add the encoder output from two directions at each time step t in the element-wise fashion to generate a new encoder output $h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$, and the final states from the inverse direction $\overleftarrow{h_1}$ and $\overleftarrow{\mathrm{C}_1}$ are sent into the decoder.

However, the inputs for the encoder for the function word sequence is not simply its word embeddings. Instead, we use the word embedding i_t at each time step to calculate the global attentions on the outputs of the encoder for the content word sequence and obtain a context vector c_t,

$$c_t = \mathrm{Attn}(i_t, o) \tag{7}$$

where $\mathrm{Attn}(\cdot)$ is defined as below.

$$c_t = \sum_{i=1}^{m} \alpha_{t,i} o_i \tag{8}$$
$$\alpha_{t,i} = \frac{\exp((e_{t,i} + g_i)/\tau)}{\sum_{j=1}^{m} \exp((e_{t,j} + g_j)/\tau)} \tag{9}$$
$$e_{t,i} = o_i^\top (\beta_t \odot \sigma(\beta_t)) \tag{10}$$
$$\beta_t = W_\beta i_t \tag{11}$$

where

$$g_i = -\log(-\log(u_i)) \tag{12}$$
$$u_i \sim \mathrm{uniform}(0,1) \tag{13}$$

Here, there are two specific details in the design of our attention mechanism. The first one is that we implement an activation function on the $e_{t,i}$, which is called Swish and proved to be effective (Ramachandran et al., 2017). The other is that we follow the idea of Gumbel Softmax to design our attention mechanism (Gumbel, 1954; Maddison et al., 2014; Jang et al., 2017). In the illustration below, the reasons why we implement such an attention mechanism here. Finally, The context vector c_t is then concatenated with the word embedding i_t and then passes through a layer of feed-forward neural network to generate the final input x_t.

Using the embeddings of the function word sequence to calculate their attentions on the outputs of the encoder for the content word sequence is reasonable in that it tends to capture the relations between each function word and the content words. The scores calculated by the attention mechanism can be regarded as a probabilistic distribution that represents the dependencies of content words on each function word, following our hypothesis proposed in the introduction. However, based on the common linguistic knowledge, in the syntactic dependency tree one function word only connects a certain number of content words, which is apparently smaller than the length of the content word sequence. The soft attention may be ineffective for the construction of the tree, while the hard attention may cause great difficulties in backpropagation. Therefore, we implement the trick of Gumbel Softmax to build a concrete distribution, which is a continuous relaxation of the discrete variables (Maddison et al., 2016). g_j is the Gumbel Noise to perturb $\log(\hat{e}_{t,i})$ to improve the robustness of the mechanism, while τ is the temperature parameter to control the degree of concreteness of the distribution, which means that the distribution becomes discrete if τ approaches zero. In our model, τ is a hyperparameter which needs to be carefully selected.

2.2 Decoder

On top of the encoder, we implement a single-layer unidirectional LSTM as decoder, which is the same as the LSTM illustrated above. The LSTM decoder to read the input words and generate summary word by word, with a fixed target vocabulary embedded in a high-dimensional space $\mathrm{Y} \in \mathrm{R}^{|\mathrm{Y}|\times \mathrm{dim}}$. At each time step, the decoder generates a summary word y_t by sampling from a distribution of the target vocabulary $\mathrm{P}_{\mathrm{vocab}}$. Different from traditional attention decoder, since we have two sequences as the input, in the decoding process, the output of the LSTM should attend to the two sequences of outputs, h and o respectively. The attention mechanism is similar to that in the encoder, and the final two context vectors are added in the element-wise fashion.

$$\mathrm{P}_{\mathrm{vocab}} = \mathrm{softmax}(\tanh(\mathrm{W_o}[c_t; s_t])) \tag{14}$$

$$s_t = \mathrm{LSTM}(y_{t-1}, c_{t-1}, s_{t-1}, \mathrm{C}_{t-1}) \tag{15}$$

$$c_t = c_t^h \oplus c_t^o \tag{16}$$

$$c_t^h = \mathrm{Attn}(s_t, h) \tag{17}$$

$$c_t^o = \mathrm{Attn}(s_t, o) \tag{18}$$

Besides, we add a memory mechanism to the attention in the decoder so that it can make use of the memory to capture deeper structural information from both the content word sequence and the function word sequence. m refers to the memory and its initialization m_0 is s_t. The memory attends to the two sequences of outputs and then adds the context vector in the element wise fashion. This process can be repeated for multiple times, and the number of times J should be set before the experiment.

$$m_{j+1} = m_j \oplus c_t \tag{19}$$

$$c_t = c_t^h \oplus c_t^o \tag{20}$$

$$c_t^h = \mathrm{Attn}(m_j, h) \tag{21}$$

$$c_t^o = \mathrm{Attn}(m_j, o) \tag{22}$$

2.3 Training

As out model is designed to be differentiable everywhere, we use backpropagation to conduct end-to-end training. Given the parameters θ and source text x, the models generates a summary $\tilde{y}$. The learning process is to minimize the negative log-likelihood between the generated summary $\tilde{y}$ and reference y:

$$\mathcal{L} = \frac{1}{\mathrm{N}} \sum_{n=1}^{\mathrm{N}} \sum_{t=1}^{\mathrm{T}} -p(y_t^{(n)} | \tilde{y}_{<t}^{(n)}, x^{(n)}, \theta) \tag{23}$$

where the loss function is equivalent to maximizing the conditional probability of summary y given parameters θ and source sequence x.

3 Related Work

Recently, neural networks are used in a wide range of tasks, including named entity recognition (He and Sun, 2017; Sun and He, 2017), word segmentation (Xu et al., 2017) and summarization (Ma et al., 2017). Additional works have also been done in exploiting the efficiency of neural networks such as sparse propagation (Sun et al., 2017; Wei et al., 2017).

Our work is also related to the sequence-to-sequence model (Cho et al., 2014b), one of the most successful generative neural model. It is widely applied in machine translation (Sutskever et al., 2014b; Jean et al., 2015; Luong et al., 2015b), text summarization (Ma et al., 2017), and other natural language processing tasks.

4 Conclusion

In this paper, we propose an end-to-end model of neural network for both machine translation and abstractive summarization, which can self-adaptively learns the paradigms of dependencies between function words and content words. The proposed method is inspired by the linguistic theory that content words in a sentence carry most of the semantic meaning of the whole sentence while function words mostly perform syntax-relevant functions. With the learned dependencies, there is no need to feed all the inputs into the RNN encoder, which may produce high computation cost. The proposed method can be naturally applied to achine translation as well as abstractive summarization. Moreover, our model contains the advantage of computation efficiency over most of the end-to-end models for machine translation and abstractive summarization.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *ACL 2004 on Interactive Poster and Demonstration Sessions*. page 31.

Joan L Bybee, Revere Dale Perkins, and William Pagliuca. 1994. *The evolution of grammar: Tense, aspect, and modality in the languages of the world*, volume 196. University of Chicago Press Chicago.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1724–1734.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. pages 1724–1734.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. pages 315–323.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. 33. US Govt. Print. Office.

Martin Hackl. 2013. The syntaxsemantics interface. *Lingua.international Review of General Linguistics.revue Internationale De Linguistique Generale* 130(130):66–87.

D. G. Hays. 1964. Dependency theory: A formalism and some observations. mem rm-4087-pr. *Language* 40(4):511–525.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI*. pages 3216–3222.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *stat* 1050:1.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*. pages 1–10.

Alexander Johansen and Richard Socher. 2017. Learning when to skim and when to read. In *The Workshop on Representation Learning for Nlp*. pages 257–264.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1412–1421.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. pages 1412–1421.

John Lyons. 1968. *Introduction to theoretical linguistics*. Cambridge university press.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*. pages 635–640.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* .

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. pages 3086–3094.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. *CoRR* abs/1710.05941. http://arxiv.org/abs/1710.05941.

F. Rosenblatt. 1960. Perceptrons and the theory of brain mechanisms. *Cornell Aeronautical Laboratory Report No* .

John Saeed. 1997. I. 2003. semantics. *GB: Blackwell Publishing* .

Xu Sun and Hangfeng He. 2017. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*. pages 713–718.

Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. 2017. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. pages 3299–3308.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014a. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014b. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. pages 3104–3112.

Leonard Talmy. 2000. *Toward a cognitive semantics, Vol. 1: Concept structuring systems.*. A Bradford Book : The MIT press.

Bingzhen Wei, Xu Sun, Xuancheng Ren, and Jingjing Xu. 2017. Minimal effort back propagation for convolutional neural networks. *arXiv preprint arXiv:1709.05804* .

Jingjing Xu, Shuming Ma, Yi Zhang, Bingzhen Wei, Xiaoyan Cai, and Xu Sun. 2017. Transfer deep learning for low-resource chinese word segmentation with a novel neural network. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*. pages 721–730.

Adams Wei Yu, Hongrae Lee, and Quoc V. Le. 2017. Learning to skim text pages 1880–1890.