INF582: Introduction to Text Mining and NLP

Lab session 5: Deep Learning for NLP

Lecture: Prof. Michalis Vazirgiannis
Lab: Antoine Tixier and Kostas Skianis

February 8, 2019

1 Introduction

In this lab, we will get familiar with the self-attention mechanism and the hierarchical architecture introduced by [1]. We will use Python 3.6 and `Keras` (version 2.2.0). Code can be found in the `main.py` script. **Note**: you must use Keras with the `TensorFlow` backend, and `gensim` version 3.2.0.[1]

2 IMDB movie review dataset

We will be performing binary classification (positive/negative) on reviews from the Internet Movie Database (IMDB) dataset[2]. This task is known as *sentiment analysis* or *opinion mining*. The IMDB dataset contains 50K movie reviews, labeled by polarity (pos/neg). The data are partitioned equally into training and testing, and into positive and negative. In the entire collection, no more than 30 reviews are allowed for any given movie, and the train and test sets contain a disjoint set of movies.

Preprocessing. We have already preprocessed the raw data, and turned each review into an array of integers of shape `(1,doc_size,sent_size)`, where `doc_size` is the maximum number of sentences allowed per document and `sent_size` the maximum number of words allowed per sentence. Shorter sentences were padded with the special padding token, while shorter documents were padded with entire sentences containing `sent_size` times the special padding token. Longer documents and sentences were truncated. The integers correspond to indexes in a vocabulary that has been constructed from the training set, and in which the most frequent word has index 2. 0 and 1 are respectively reserved for the special padding token and the out-of-vocabulary token.

Binary classification objective function The objective function that our model will learn to *minimize* is the *log loss*, also known as the *cross entropy*[3]. More precisely, in a binary classification setting with 2 classes (0 and 1), the log loss is defined as:

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\big(y_i \text{log} p_i + \big(1-y_i\big)\text{log}\big(1-p_i\big)\big) \tag{1}$$

3 Self-attention

The attention mechanism [7] was developed in the context of encoder-decoder architectures for Neural Machine Translation [8, 3], and rapidly applied to related tasks such as image captioning

[1] Use `pip show gensim` to find out which version you're using. If needed, install the right version with `pip install gensim==3.2.0`

[2] http://ai.stanford.edu/~amaas/data/sentiment/

[3] https://keras.io/objectives/

(translating an image to a sentence) [4], and summarization (translating to a more compact language) [5]. Attention devices have also been proposed for encoders only [1, 2]. Such mechanisms are qualified as *self* or *inner* attention. We briefly present them next.

Consider a RNN encoder taking as input a sequence of vectors $(x_1, \ldots, x_T)$ of length T. By definition, the RNN maps the input sequence to a sequence of hidden representations $(h_1, \ldots, h_T)$ of length T. The hidden representations are often called *annotations* in the literature. Rather than considering the last annotation h_T as a summary of the entire sequence, which is lossy (since h_T is a single vector), the self-attention mechanism computes a new hidden representation as a weighted sum of the annotations at *all* time steps.

Eq. 2 illustrates self-attention in its most simple variant. The annotations (h vectors) are first passed to a dense layer. The alignment coefficients (stored in the α vector) are then computed by comparing the output of the dense layer with a trainable context vector u and normalizing with a softmax. The attentional vector s is finally obtained as a weighted sum of the annotations.

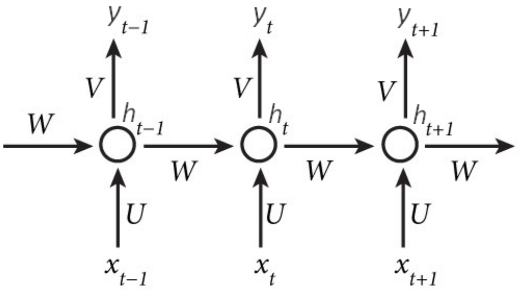

Figure 1: 3 steps of an unrolled RNN.

$$u_t = \tanh(Wh_t) \qquad \alpha_t = \frac{\exp(u_t^\top u)}{\sum_{t'=1}^{T} \exp(u_{t'}^\top u)} \qquad s = \sum_{t=1}^{T} \alpha_t h_t \tag{2}$$

The context vector can be interpreted as a representation of the optimal element[4], on average. When faced with a new example, the model uses this knowledge to decide how much it should pay attention to each element in the sequence. The context vector is initialized randomly and updated during training.

4 Hierarchical Attention

An interesting application of self-attention was provided by [1] and is illustrated in Fig. 2. In this architecture, the self-attention mechanism comes into play twice: at the word level, and at the sentence level.

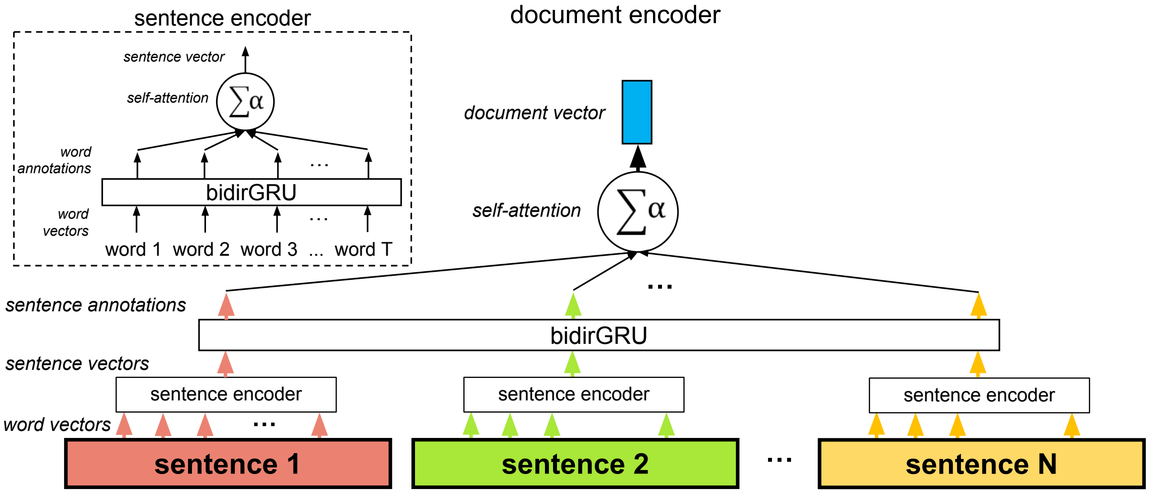

Figure 2: Hierarchical Attention Network.

[4] elements can be characters, morphemes, words, sentences...

The HAN architecture makes sense for two reasons: first, it matches the natural hierarchical structure of documents (words → sentences → document). Second, in computing the encoding of the document, it allows the model to first determine which words are important in each sentence, and then, which sentences are important overall. Through being able to re-weigh the word attentional coefficients by the sentence attentional coefficients, the model captures the fact that a given instance of a word may be very important when found in a given sentence, but another instance of the same word may not be that important when found in another sentence.

Question (just as an in-class activity - you don't have to submit any answer to Moodle).
- what are some limitations of the HAN architecture?

References

[1] Yang, Zichao, et al. "Hierarchical attention networks for document classification." Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.

[2] Lin, Zhouhan, et al. "A structured self-attentive sentence embedding." arXiv preprint arXiv:1703.03130 (2017).

[3] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

[4] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).

[5] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).

[6] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[7] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[8] 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.