Research paper reading

A simple but tough-to-beat baseline for sentence embeddings

CHRIMNI Walid

DS Telecom - Natural Language Processing

November 2021

Sommaire

Introduction

After implementing word embeddings a decade ago, researchers are trying to capture the semantic meaning of groups of words (sentences, paragraphs,...). Several methods have been developed, ranging from the very complicated to the very simple : composing word embeddings using operations on vectors and matrices ; unweighted averaging of word embeddings, neural netword structures such as RNN and CNN... This paper presents a simple, yet very effective method for sentence embeddings.

1 Explanation

The method used is an improvement of an already existing method developed by Wieting et al. in 2016. Wieting's embeddings were based on even simpler method : retraining of word embeddings and basic linear regression. The main problem of this approach is that it requires retraining with a substantial labeled dataset. This led the paper writers to improve the method. This new method beats sophisticated models such as RNN and LSTM.

As the method is based on Wieting et al.'s, we will start by explaining their method. The principle is simple : they got a standard word embeddings and modified them based on supervision from the Paraphrase pairs dataset. Then, they construct sentence embeddings by training a simple word averaging model. This allows to create a model where embeddings encapsulate the general meaning of a sentence, since it was trained on paraphrase. As Wieting et al. report that simple average of the initial word embedding does not work very well, it seems like the keystone of this approach is the supervised training on the paraphrase dataset, which can be quite the restriction.

This paper's method is based on *smooth inverse frequency* which is a weight given a word w : $\frac{a}{a+p(w)}$ where a is a parameters and $p(w)$ the estimated word frequency. $p(w)$ is It is very simple to compute the sentence embeddings : get a word embeddings, compute the weighted average with the SIF weighting and then remove the projections of the average vectors on their first singular vector. This elementary method allows to beat the unweighted average on a variety of textual similarity tasks, and it even achieves better performance than sophisticated model in most of these tasks.

The SIF weighing does not come out of nowhere, the paper give a theoretical origin. It comes from an improved Random Walk model of (Arora et Al., 2016) : the discourse vector c_t is assumed not to change much so it may be considered constant. Thus, given the discourse vector c_s, the probability that the word w is emitted in the sentence s can be written :

$$\Pr[w \text{ emitted in sentence } s \mid c_s] = \alpha p(w) + (1-\alpha)\frac{\exp\left(\langle \tilde{c}_s, v_w \rangle\right)}{Z_{\tilde{c}_s}}$$
$$\text{where } \tilde{c}_s = \beta c_0 + (1-\beta) c_s, c_0 \perp c_s$$

where α and β are scalar hyperparameters, and $Z_{\tilde{c}_s} = \sum_{w \in \mathcal{V}} \exp\left(\langle \tilde{c}_s, v_w \rangle\right)$ is the normalizing constant.

Here the *smooth* term in SIF becomes clear : this model has two type of smoothing term : the first one accounts for the fact that some words occur out of context ; the second one accounts for frequent words (such as stop words) that appear often regardless of the discourse.

Then, given the discourse vector c_s, the probability of a sentence s to appear is the product of the probability of all its word w. The gradient of the log of this quantity is then easily calculated. Thus, the MLE of $\tilde{c_s}$ is proportional to the sum of the SIF weighs : $\tilde{c}_s \propto \sum_{w \in s} \frac{a}{p(w)+a} v_w$, where $a = \frac{1-\alpha}{\alpha Z}$. To obtain the final sentence embedding, one juste have to substract the projection of $\tilde{c_s}$ to their first principal component, that is subtracing c_0.

The reason why they modifed Arora et al.'s theoretical model is that empirically, we observe that most word embedding methods give large vectors to frequent words which causes the average of word vectors to have huge components along semantically meaningless directions.

2 Advantages and issues of the method

The method has two main, clearly identified advantages : it is simpler than other methods (simple to compute and unsupervised) and it beats other methods. Moreover, it is similar to Wieting et al.'s method it not longer needs a labelled dataset for training since it uses unsupervised training.

The method is well-suited for domain adapdation settings : word vectors trained on various corpora are used to compute the sentence embeddings. The performance of the model does not seem to be affected by the corpora the word frequencies $p(w)$ are calculated from, which make it a very versatile model. One could think of calculating the word frequencie on different (and specific) datasets for certain tasks. For instance, if one aims to create a model for textual similarity tasks with medical corpora, computing the $p(w)$ on medical texts may improve the performance. Similarly, a wide range of the parameters a can achieve close to best results.

An other advantage that we can see for this model is his theoretical backbone. Even though it was motivated by empirical observations, we have the proof (under some assumptions) that the model theoretically works well. The last advantages we could note is the versatility of the model : the method can be applied to any types of word embeddings.

As for the issues of the model, the first one is that it works well only on a definite

set of tasks. As result, it does not perform well for sentiment analysis tasks for two main reasons :

- **the antonym problem :** the model suffers from the antonym problem, it struggles to associate opposite representation to words that are semantically opposite not in the naive way ("bad" and "good" for instance)
- **the weighted average scheme :** it entails word like "not" to be downweighted a lot, while they are important for sentiment analysis.

Moreover, when running the code that was provided, we realized it was very time-consuming to complete sentences embeddings. It is a pity that they did not talk about memory or time consumption.

The hypothesis that c_t does not change much over time can also be discussed. We could arguably say that it is not always true, especially for long sentences where the topic can change between the beginning and the end.

3 My personnal take

To me, the topic is quite interesting. I think sentence embedding was quite new at the time (2017), so it may have been even exciter then. In any case, i find it interesting to try to capture semantic senses of group of words instead of words themselves. Even though other (and similar) method already existed at the time the paper was released, it is quite surprising to see that such a simple method gives such good results. The paper is perfectly in tune with its time, with the issues of its time. Even if it may be outdated today (as we will see in the next section).

I find it great how the researchers of the paper derived most of their result from empirical observation and afterwards try to justify it theoretically. I feel like it is generally done the other way around, so this way of doing is nice.

I like how the paper uses other existing work to make better and simpler model. In my opinion, that how researchers should work.

As for the topic in general, I think it is one of the most fascinating subject, even nowadays. It is funny to see to see the parallel between our comprehension and that of the machine : we can easily understand sentences and capture semantic senses but we can't proceed many sentences quickly ; whereas it is hard to make the machine understand the senses and subtlety of sentences but it can proceed sentences very quickly.

To improve the work, we could think of a model that take into account the order of words in sentences. In some cases, it is a pro that the model does not take it into account, but for in other cases it is the other way around. It would be nice to be able to choose whether to take the order of words into account or not.

4 Related work and posterity

The method was on the same level or better than most sentence embeddings model at the time. (Mitchell Lapata, 2008 ; 2010 ; Blacoe Lapata, 2012) used vectors and matrices operation to combine word embeddings and create sentence embeddings. (Mikolov et al., 2013a) found out that unweigheted average performed quite well in representing short phrases. These methods were very simple but did not gave as good results as complex methods. As for complex methods (using RNN, LSTM, CNN...) performed well but were a bit complex. The model introduced in this paper combines the advantages of both.

Nevertheless, the model does not seem to have been a revolution, nor to have been followed on that much. Since that time, better models have emerged, in particular model based on transformers. This is what may have caused this. The paper "Attention Is All You Need", release at about the same time, may have masked the paper. It is based on Transformer. It is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. Transformer is to handle the dependencies between input and output with attention and recurrence completely. The library « Hugging Face » provides natural language processing technologies based on Transformers. Examples showing that Hugging Faces' models achieves better performance are available in the notebook.