Evaluating Sentence Representations

Joris Mollinga
University of Amsterdam
Science Park 604
joris.mollings@student.uva.nl

Aron Hammond
University of Amsterdam
Science Park 604
hammond756@live.nl

Abstract

In this paper we compare the Skip-gram and Embed-align model on a variety of NLP tasks using the SentEval toolkit. In particular, we compute the sentence embeddings using three methods; an average, a weighted average on the importance of words in a sentence and a smoothed inverse frequency average. We show that the use of more sophisticated averaging method does not have a big effect on performance, and that the results of the Embed-align and Skipgram model are comparable, with minor differences on some tasks[1].

1 Introduction

Most state of the art Natural Language Processing (NLP) systems rely on dense vector representations, word embeddings, to encode meaning. These embeddings can easily be calculated with various Neural Network architectures.

However, for most NLP tasks and applications one is interested in the meaning of a sentence and the way words interact instead of individual words. Various techniques exist to turn word embeddings into a sentence embedding. The simplest method is to average over all word embeddings in a sentence, but this method performs poorly on long sentences. Next to a simple average we use two different methods to calculate a weighted average of word embeddings in a sentence, called the Smooth Inverse Frequency [1] and InferSent Model [3].

[1]Code available at: https://github.com/jorism1993/ULL-Practicals

Until recently, different models were compared using their own baseline and comparing different models was a difficult task. Until recently, when SentEval [2] was introduced. SentEval is a toolkit for evaluating sentence embeddings across a variety of tasks, including analysis of subjectivity/objectivity, semantic relatedness and paraphrase detection. In this paper we compare three different ways of computing sentence embeddings on the Skip-gram and Embed-align, and evaluate their performance using the SentEval toolkit.

First, we will try to place this work in it's academic context. In section 3 we explain the models and averaging methods under evaluation. We also provide an overview of the evaluation framework in section 4 and finally try to explain the results in section 5.

2 Previous work

Mikolov et al. [4] sparked a wave of research into deep neural networks for word embeddings by introducing an efficient and effective method to produce dense word embeddings. Models trained on huge corpora have shown to have desirable properties and good performance on extrinsic evaluations. There have been many adaptations of this method. Embed-align [6] uses sentence aligned corpora to learn word embeddings that can disambiguate word meaning based on context. Works like [8], [1] and [3] have taken these models and shown how the individual word embeddings can be combined to produce sentence embeddings with the same ability to generalise across tasks. There are two different ways in which researchers are trying to achieve this: su-

pervised, like [3] or unsupervised, like [1]. This paper will compare the unsupervised approaches to this problem.

3 Methods

3.1 Computing word embeddings

The models analysed in this paper used to compute word embeddings are Skip-gram [4] and Embed-align [6]. Both models try to solve the problem of producing compact representations of word meaning but approach this problem differently.

Skip-gram is essentially a discriminative classifier that tries to predict whether a word co-occurs with a another word in its context. We copied a publicly available Skip-gram implementation [2] and trained it on the English Europarl corpus using a window size of 2.

Embed-align is a generative model that jointly learns parameters for a Gaussian distribution to represent each word in a given context and its corresponding word alignment in a foreign language corpus. The main difference is that Embed-align produces different vectors based on the context a word is in whereas Skip-gram learns an average of all contexts. Both models have an embedding dimension of 100. If a word in a sentence is unknown by one of the models analysed in this paper, it is mapped to a vector of zero's. This is preferred to mapping a word to *UNK*, because taking a weighted average of an unknown word seems counter intuitive.

3.2 Computing sentence embeddings

This paper focuses on the performance of various unsupervised methods of combining word embeddings into sentence embeddings. The range of options is limited by the SentEval interface because the evaluation sentences can only be accessed in batches while some methods, like applying singular value decomposition (SVD) require pre-processing the entire corpus. We compare three different ways of computing a sentence embedding from word embeddings. A simple average and two types of weighted averages.

1. **Average**
   As a baseline for the composition methods we use the the average of all the words in the sentence. Although it is very simple it has been shown to work well on a variety of downstream tasks [8]. The disadvantage of taking the average of more than a few words is obvious: we end up with a point in the middle which has lost the meaning of the sentence. The other two methods try to circumvent this problem.

2. **Smooth Inverse Frequency**
   To incorporate information about relative word importance in the sentence embeddings, we implemented a simplified version of smooth inverse frequency (SIF) as described in [1]. This method calculates the sentence embedding v_s for sentence S by weighting the word embeddings v_w by a function of their unigram probability $p(w)$ in the corpus and a hyper parameter a:

$$v_s = \frac{1}{|S|} \sum_{w \in S} \frac{a}{a + p(w)} v_w$$

   Based on the reported performance in [1], the value for a has been fixed to 0.01. The equation above is derived from a random walk model. For a detailed derivation we refer the reader to the original paper. The main effect of this transformation is that words that occur less in the corpus get assigned a higher weight, implementing the assumption that those words carry more meaning. In the original algorithm PCA is performed on a matrix containing all sentence embeddings and the projection of v_s onto the first principal component is subtracted from itself. Due to the interface with SentEval we were not able to implement this.

3. **InferSent importance weights**
   InferSent [3] is a method to calculate sentence embeddings that is trained on natural language inference data and generalises well to many applications. The InferSent model is capable of generating a distribution over words in a sentence that captures their importance. See Figure 1 for an example. The InferSent model clearly assigns *Pancakes*, *favourite* and *food* as the most important words.

[2] https://github.com/fanglanting/skip-gram-pytorch

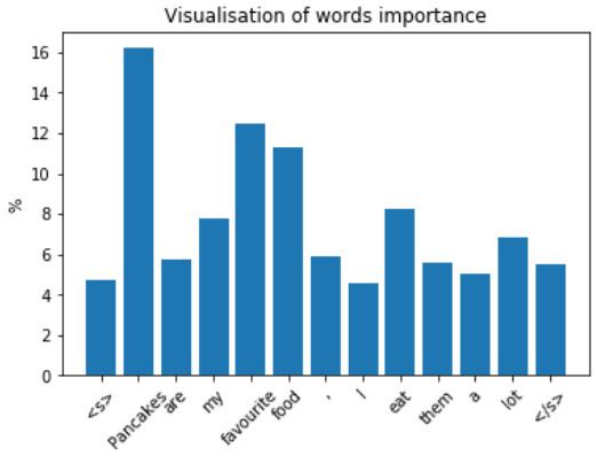

Figure 1: Visualisation of word importance on the sentence assigned by InferSent: *Panackes are my favourite food, I eat them a lot*

Although the InferSent model is capable of generating a sentence embedding for any given sentence, we only use the feature that generates a word importance distribution. This distribution is then used to calculate a weighted average of word vectors.

The InferSent model uses all the GloVe word embeddings (approximately 2.2M words) to determine the word importance distribution.

4 Evaluation

The methods outlined in the previous section will be evaluated on nine downstream task. Researchers at Facebook AI Research (FAIR) have published a toolkit called SentEval that affords others to easily and reliably evaluate their sentence embeddings [2]. This section will briefly describe the tasks considered in our experiment. The tasks are performed by simple models like a Multi-layer perceptron (MLP) or logistic regression using the generated sentence embeddings as features.

4.1 Binary Sentiment

Four tasks involve a binary classification task in which a model tries to predict whether a sentence was either positive or negative. These are: customer reviews (CR), movie reviews (MR and SST-2) and opinion polarity (MPQA).

4.2 Mutli-class Classification

The TREC task consists of labelling a question with one of five question types.

4.3 Subjectivity

A binary classification task (SUBJ) where a model has to classify a sentence as either subjective or objective.

4.4 Natural Language Inference

One task (SICK-E) with the objective to label two sentences as entailing, contradictory or neutral with respect to each other.

4.5 Paraphrase Detection

The MRPC task requires a model to classify a sentence pair as either a paraphrase or not.

4.6 Semantic Textual Similarity

The STS14 task evaluates the similarity of two sentences on a continuous scale and compares this with a human labelled similarity score.

5 Analysis

The results of the evaluation can be found in Table 1. In general, the Embed-align model performs slightly better than the Skip-gram. This is probably because the word embeddings produced by Embed-align already encode contexts and are therefore better at discriminating sentence meaning. However, it is worth noting that the performance doesn't vary much for a particular task, a few tasks excepted. The analysis consist of a set of hypothesised explanations that were formed on the basis of our prior knowledge about the properties of distributional semantic models.

The first exception is the TREC task, where Skip-gram greatly outperforms Embed-align. An explanation for this would be the small window size on which the SG model was trained. The TREC tasks involves identifying the objective of a question (eg. the questions asks for a calendar date). A small window size would cause the model to capture more syntactic information in the word embeddings which in turn might help identify the type of question that is being asked.

Table 1: Evaluation results of Skip-gram (SG) and Embed-align (EA) with different composition methods on a selection of downstream tasks. The best performing combinations of model and composition per task are in bold. * First number is pearson correlation, second number is spearman correlation. ** EA result for STS14 taken from [6]. N.A. due to technical difficulties.

	CR	**MPQA**	**MR**	**MRPC**	**SICK-E**	**SST2**	**SUBJ**	**TREC**	**STS14***
SG	**72.71**	81.72	65.4	70.32	74.65	66.28	**84.27**	**78.6**	0.39/0.41
SG + SIF	72.31	81.79	65.51	69.33	72.97	67.05	83.78	78.2	0.45/0.46
SG + IN	72.29	82.1	64.57	68.64	73.68	64.63	82.46	78.0	0.43/0.44
EA	70.46	83.77	64.77	71.01	**74.81**	67.27	79.16	57.0	**0.69/0.59****
EA + SIF	69.96	84.01	64.48	**71.65**	72.64	**67.66**	79.8	56.4	N.A.
EA + IN	71.95	**84.84**	**66.27**	68.81	70.59	66.94	81.34	62.6	N.A.

The second task where the Skip-gram outperforms the Embed-align is the subjectivity task. A possible explanation for this could be that the difference is style between the training corpus (Europarl) and evaluation sentences. Europarl contains very dry, boring, sentences while the subjectivity task involves more emotive, prozac and even sarcastic sentences. This could throw off the word embeddings produced by Embed-align because the Bi-LSTM wasn't trained on encoding these types of sequences. Skip-gram was also trained on Europarl, but doesn't dynamically produce word embeddings during evaluation like Embed-align does.

In the opinion polarity task (MPQA) we attribute the high performance of the EA + IN model to the short length of the sentences. The average sentence length in the MPQA task is 3.1 words. Redistributing weights across word vectors has the biggest influence in short sentences, hence the high score.

Another difference between Skip-gram and Embed-align is that Skip-gram doesn't seem to benefit from any form of weighted averaging whereas the performance of Embed-align is improved by at least one of the modifications for most tasks. From this we can conclude that taking weighted averages works best when the word embeddings capture more context.

6 Conclusion

Overall the use of more sophisticated averaging methods doesn't seem to to do much good or bad for the performance while being slightly more complex to implement and in the case of InferSent weighting also drastically increasing the run time. Especially the baseline Skip-gram model does not benefit from other averaging method. This, in combination with the fact that Embed-align outperformed Skip-gram, suggests that improving the performance of sentence embeddings on downstream task might not be achieved by means of combining the word embeddings. Progress could be better made by focusing on expressiveness by using more advanced methods of learning word embeddings, training them on a larger corpus or using word embeddings trained for a specific task.

7 Future work

In this experiment we have only considered composition techniques that are some form of averaging word vectors. It will be interesting to expand this research into entirely different approaches. Examples of these are parse-tree based composition [7] and Recurrent Neural Networks [5]. It could be that these more advanced methods do improve composition beyond the level of simple averaging. Another note is that the differences in accuracy were not tested for statistical significance so the small differences could also be due to random chance. In the future it would be good to incorporate this information in the results generated by SentEval. Furthermore it would be interesting to analytically investigate the proposed causes for the difference in performance by probing the models and zooming in on particular failure modes.

References

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[2] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[3] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

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

[5] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *CoRR*, abs/1502.06922, 2015.

[6] Miguel Rios, Wilker Aziz, and Khalil Sima'an. Deep generative model for joint alignment and word representation. *arXiv preprint arXiv:1802.05883*, 2018.

[7] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[8] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.