Unsupervised Language Learning - Practical 3

Nuno Mota
11413344

Tom Pelsmaeker
10177590

Abstract

In this short paper we evaluate the sentence embeddings produced by a SkipGram model (Mikolov et al., 2013b) and ones produced by an EmbedAlign model (Rios et al., 2018). The evaluation is performed on Facebook's SentEval framework (Conneau and Kiela, 2018). Overall, our results show that EmbedAlign is the most powerful of the two, in most tasks.

1 Introduction

Natural Language Processing (NLP) is a task that is inherently based on a discrete domain, i.e., words and punctuation. Without knowledge about the meaning and usage of each individual word, it becomes hard for computers to work on the language domain. It would, then, be preferable to have a word representation (for each word) that allows the computer to better reason about language tasks. Additionally, most powerful current methods require the data domain they are applied on to be continuous. These two ideas smoothly lead to the concept of representing words in a vectorial, or even distributional, manner, i.e. word embeddings.

In this article we focus on the embeddings produced by the SkipGram model (Mikolov et al., 2013b) and the EmbedAlign model (Rios et al., 2018).

SkipGram attempts to learn embeddings by taking into account the context (a window) of a word. It tries to maximise the context words categorical probability, conditioned on the centre word, over the entire dataset.

On the other hand, EmbedAlign makes use of translation data (that is, a parallel corpus), instead of using only monolingual data. It attempts, starting from the source language, to produce a distributional embedding common to both the source and target language, from which it has to be able to reconstruct both languages.

The results of our experiments indicate that EmbedAlign produces more informative embeddings, outperforming SkipGram on most of the used SentEval tasks (Conneau and Kiela, 2018).

Both the models and the findings will be discussed more in depth in the next sections[1].

2 Background

A word representation is useful when it captures informative properties of the word, such as its meaning and syntactics. The distributional semantics hypothesis (Harris, 1954) (Firth, 1957), claims that word context is informative to the properties of a word. Following this hypothesis, a word embedding is a representation of a word as a dense real-valued vector (or distribution), constructed from its context. What sets different methods of obtaining word embeddings apart is how they treat the context of a word to obtain its embedding. Such design choices have impact on the performance of word embeddings on downstream tasks.

Traditionally, such embeddings were created from global co-occurrence statistics within a corpus, for instance by approximate factorization of the word-context co-occurrence matrix into lower dimensional word and context embeddings (Baroni et al., 2014). Well known early models of this type are LSA (Landauer and Dumais, 1997) and its extension to word-word matrices HAL (Lund and Burgess, 1996). Whilst these models work quite well for small amounts of data, they generally struggle with large corpora (Pennington et al., 2014).

Neural word embeddings first arose as a byproduct of neural language models (Bengio et al., 2003), which often embed one-hot vectors to lower-dimensional representations as the first layer of a deeper network. Yet, these embeddings were not specifically designed to capture the distributional properties of language, and training did not scale well to large corpora because of the deep architectures. Addressing these issues, (Mikolov et al., 2013a) designed two shallow models specifically designed to embed words based on their context, SkipGram and CBOW (together Word2Vec). The models learn word embeddings from local context by either predicting context from centre (SkipGram) or centre from context (CBOW). As such, it learns from *local* co-occurrence statistics within a corpus. Word2Vec scales well to large corpora and generally creates better embeddings than factorization methods (Baroni et al., 2014).

[1]Evaluation code available here: `https://github.com/0Gemini0/ULL/tree/master/Practical3`

Word2Vec spurred a breadth of research on word embeddings. Glove (Pennington et al., 2014), for example, is a global log-bilinear regression model that produces word embeddings by essentially performing a matrix factorization on the nonzero elements of a global cooccurrence matrix, combining insights from Word2Vec with earlier factorization methods. Dependency based word embeddings (Levy and Goldberg, 2014) are an adaptation of Skipgram where context stems from a window over dependency graphs instead of raw sentences, yielding embeddings that capture similarity over relatedness. Bayesian SkipGram (Bražinskas et al., 2017) is another adaptation of SkipGram that encodes embeddings as distributions in a latent space instead of point estimates, allowing the model to capture multiple senses of the same word. Recently, EmbedAlign (Rios et al., 2018) proposes to learn embeddings as distributions based on multilingual contexts.

None of the models described above can be said to conclusively produce the *best* embeddings overall, all claiming superiority in at least some task or metric. This highlights the need for a standardized benchmark for embeddings. The SentEval framework (Conneau and Kiela, 2018) aims to be precisely this, and will be discussed in Section 4. In the next section SkipGram and EmbedAlign will be discussed in more detail.

3 Method

In this section we explain in detail both models and the specifics to our implementation.

3.1 SkipGram

For a given centre word, SkipGram tries to predict the probabilities of a categorical that is assumed to model the distribution of possible context words. The model is then trained by estimating this categorical parameters for each token in the dataset (possibly many times, over several epochs). Each token will, at some point, be considered as a centre word and have a corresponding set of context words, determined by a window size.
Given a corpus T, the training objective of the model is to maximize the log-probability of context prediction for each word in T. That is:

$$\underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{argmax}\left(\sum_{w\in T}\sum_{c\in C_w}\log\left(p_{\boldsymbol{\theta}}\left(c|w\right)\right)\right) \tag{1}$$

where $\mathcal{C}_w$ the set of context words for w and $p_{\boldsymbol{\theta}}\left(c|w\right)$ the corresponding estimated categorical probability for context word c given centre word w. In the case of SkipGram $p_{\boldsymbol{\theta}}\left(c|w\right)$ simply takes the form of a softmax applied to the inner product of the centre word embedding (from centre words' embeddings' matrix ν) and context word embeddings (from context words' embeddings' matrix ν'):

$$p_{\boldsymbol{\theta}}\left(c|w\right)=\frac{\exp\left(\vec{\nu_w}^{\,T}\cdot\vec{\nu_c}'\right)}{\sum_{k=1}^{|V|}\exp\left(\vec{\nu_w}^{\,T}\cdot\vec{\nu_{c_k}}'\right)} \tag{2}$$

The computation of the softmax can be extremely expensive, for large vocabulary sizes ($|V|$).
Fortunately, tractable approximations exist, such as Negative Sampling. Let us consider that, due to independence given the parameters, our training instances are in fact pairs of (centre word - context word) over which we try to maximize (1). The idea of Negative Sampling is to take advantage of that, create pairs that do not occur in the data and train the model by having it assign a high probability to positive (real, in the data) pairs and assign a low probability to negative (fake, not in the data) pairs. In essence, our training objective is modified to:

$$\begin{aligned}\underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{argmax}\,(\,&\sum_{(w,c)\in\mathcal{P}}\log\left(p_{\boldsymbol{\theta}}\left((w,c)\in\mathcal{P}\right)\right)+\\&\sum_{(w,c')\in\mathcal{N}}\log\left(1-p_{\boldsymbol{\theta}}\left((w,c')\in\mathcal{P}\right)\right)\,)\end{aligned} \tag{3}$$

where $\mathcal{P}$ is the set of positive pairs, $\mathcal{N}$ is the set of negative pairs and

$$p_{\boldsymbol{\theta}}\left((w,c)\in\mathcal{P}\right)=\frac{1}{1-\exp\left(\vec{\nu_w}^{\,T}\cdot\vec{\nu_c}'\right)} \tag{4}$$

Note that given the independence of centre tokens to one another, we can use some kind of stochastic optimisation (Robbins and Monro, 1951).

3.2 EmbedAlign

Unlike SkipGram, EmbedAlign produces embeddings that are probability distributions, which are learnt from lexical translation data as an indirect semantic supervision signal. For any two languages, L1 being the language of interest and L2 an auxiliary language, some assumptions are made, namely: every word in one of the languages has a word of equivalent meaning in the other language; these equivalences can be found by using lexical alignments; at training time the length of all sentences is known. The main idea of EmbedAlign, which is a deep generative model, can be described as follows:

For each sentence pair (x_1^m, y_1^n), where x_1^m represents the L1 sentence, of length m, and y_1^n represents the L2 sentence, of length n, we get an intermediate encoding, $\vec{h}_1^m$, for each of the L1 sentence's words. This

intermediate encodings can be obtained through several different methods. Such as using deterministic (but trainable) encodings or using the average hidden state per word of the forward and backward pass through a BiRNN. The latter has the advantage of each encoding acquiring some of the sentence's context.

Each element of $\vec{h}_1^m$ is then used to estimate, using feed forward Neural Networks, the mean, $\vec{u}_1^m$, and diagonal standard deviation, $\vec{s}_1^m$, parameters of a Gaussian distribution that is meant to approximate the posterior distribution of latent embeddings $\vec{Z}_1^m$. Supposedly, all elements of $\vec{h}_1^m$ would be used to produce the parameters of the distribution that would encode the entirety of $\vec{Z}_1^m$. However, this approach would lead to an intractable marginalisation over the latent embeddings. Applying a mean field approximation circumvents this, resulting in the approximate posterior q_ϕ:

$$q_\phi\left(\vec{z}_1^m|x_1^m\right) = \prod_{i=1}^{m} \mathcal{N}\left(\vec{z}_i|\vec{u}_i, \texttt{diag}\left(\vec{s}_i \odot \vec{s}_i\right)\right) \tag{5}$$

As such, each word embedding is now generated independently from the encodings of the other words. From the estimated means, $\vec{u}_1^m$, and diagonal standard deviations, $\vec{s}_1^m$, samples $\vec{z}_1^m$ of $\vec{Z}_1^m$ are obtained. The reparameterisation trick of (Kingma and Welling, 2013) is used when sampling to allow for backpropagation through the samples. The samples $\vec{z}_1^m$ are then passed through two independent NNs, with softmax activations on the last layer, in order to produce the parameters of the categorical distributions, $P_\theta\left(x_i|\vec{z}_i\right)$ and $P_\theta\left(y_i|\vec{z}_i\right)$. From these we can evaluate (or sample) the training pair (x_1^m, y_1^n).

Since we perform variational inference in order to approximate the true posterior $p\left(\vec{z}_1^m|x_1^m\right)$, we have to maximise the parameters on the Evidence Lower Bound (equation 6). Note that, since we do not know to which $\vec{z}_i$ each of the y_j corresponds to, the alignments are marginalised, with a uniform probability over the alignments $P\left(a_j|m\right)$.

$$\begin{aligned}
&\log\left(P_\theta\left(x_1^m, y_1^m|m,n\right)\right) \geq \sum_{i=1}^{m} \mathop{\mathbb{E}}_{\vec{z}_1^m \sim q}\left[\log\left(P_\theta\left(x_i|\vec{z}_i\right)\right)\right] + \\
&\sum_{j=1}^{n} \mathop{\mathbb{E}}_{\vec{z}_1^m \sim q}\left[\log\left(\sum_{a_j=1}^{m} P\left(a_j|m\right) P_\theta\left(y_j|\vec{z}_{a_j}\right)\right)\right] - \\
&\sum_{i=1}^{m} \mathrm{KL}\left(\mathcal{N}\left(\vec{z}_i|\vec{u}_i, \texttt{diag}\left(\vec{s}_i \odot \vec{s}_i\right)\right) || \mathcal{N}\left(\vec{z}_i|\vec{0}, \mathbf{I}\right)\right)
\end{aligned} \tag{6}$$

Again, let us note that the computation of the softmax, for both $P_\theta\left(x_i|\vec{z}_i\right)$ and $P_\theta\left(y_i|\vec{z}_i\right)$, can be intractable for large vocabularies. In the case of EmbedAlign we approximate the softmax computation directly, by using Complementary Sum Sampling (CSS) (Botev et al., 2017). The idea of CSS is simply to obtain an estimate for the normaliser. Instead of computing it completely, a negative set of words, akin to SkipGram, is sampled for each mini batch. Specifically, the negative set of words, $\mathcal{N}$, is a subset of all words not found in the mini batch, while the positive set consists of all the words in the mini batch. Note that this is equivalent to applying the softmax on only some elements of the output of the NNs previously described (the ones that generate $P_\theta\left(x_i|\vec{z}_i\right)$ and $P_\theta\left(y_i|\vec{z}_i\right)$). Each element of the output, before the softmax, can be seen as a scoring function $u_\theta\left(\vec{z}, w\right)$ between embedding z and word w, allowing us to approximate the softmax $P_\theta\left(w|\vec{z}\right)$ with:

$$\frac{\exp\left(u_\theta\left(\vec{z}, w\right)\right)}{\sum_{pw \in \mathcal{P}} \exp\left(u_\theta\left(\vec{z}, pw\right)\right) + \sum_{nw \in \mathcal{N}} \kappa\left(nw\right) \exp\left(u_\theta\left(\vec{z}, nw\right)\right)} \tag{7}$$

where $\kappa\left(nw\right)$ corrects for bias as $\mathcal{N}$ tends to the entire complement set ($\mathcal{W} \backslash \mathcal{N}$, where $\mathcal{W}$ is the set of words in the language).

3.3 Model Training

Both models were trained with the Europarl parallel en-fr corpus (Koehn, 2005). Skipgram, with negative sampling, was trained with settings comparable to (Rios et al., 2018). Frequent words were not sub-sampled, nor was the vocabulary size reduced. However, all tokens were lowercased. We used a context window of 5, padding when necessary, and sampled, according to unigram statistics raised to the power of $\frac{3}{4}$ (Mikolov et al., 2013a), as many negative context words as we had positive ones (k=1). Both models embed into $\mathbb{R}^{100}$.

SkipGram was trained until convergence on training loss with the Adam Optimizer (Kingma and Ba, 2014) with learning rate of $1e-2$ and batch size of 1024, per the linear scaling rule (Goyal et al., 2017). Early stopping based on validation loss was not applied as it is a bad predictor of task performance (Lai et al., 2016).

4 Evaluation

We assess the quality of the models based on their capacity to produce informative sentence representations, using the SentEval framework (Conneau and Kiela, 2018) as benchmark. Specifically, we report the classification accuracy (unless otherwise specified) on the following downstream NLP-tasks using sentence representations induced by the models as features:

MR Classifying movie reviews as positive/negative.
CR Classifying product reviews as positive/negative.
SUBJ Classifying sentences as subjective/objective.
MPQA Classification of opinion polarity.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-E	STS14
SG_{AVG}	64.9	70.6	**79.8**	83.6	64.3	53.8	70.1/80.2	67.7	0.46/0.48
SG_{TF-IDF}	64.2	69.2	77.5	83.3	64.8	48.8	69.7/80.1	64.5	0.50/0.51
EA_{AVG}	64.7	70.7	79.2	83.8	**67.1**	**57.4**	71.0/80.1	**74.75**	0.58/0.57
EA_{TF-IDF}	64.4	69.9	77.9	83.8	**66.7**	50.2	**71.5/80.8**	73.0	**0.63/0.61**

Table 1: Results on Various SentEval task (see section 4) of SkipGram and EmbedAlign with averaged word embeddings and tf-idf weighed averaged word embeddings. The best result per task that we assume to be significant is boldfaced.

SST Fine grained sentiment analysis of movie reviews.
TREC Classifying questions over multiple classes.
MRPC Detection of paraphrases (accuracy/F1).
SICK-E Classifying textual entailment.
STS14 Semantic textual similarity (Pearson/Spearman).

As the models infer word embeddings, a method is required to aggregate the embeddings into a sentence embedding. The simplest method is averaging all the word vectors in a sentence. However, not all words in a sentence might be of equal importance. To capture this, a weighted average of word embeddings can be taken based on their TF-IDF score (Salton and Buckley, 1988):

$$\text{TF-IDF}_{w/s} = \frac{f_{w/s}}{|s|} \cdot \frac{|S|}{|\{s \in S : w \in s\}|} \quad (8)$$

Where s is a sentence, w a word, S the set of sentences in a task and $f_{w/s}$ the frequency of words in a sentence, yielding the $\text{TF-IDF}_{w/s}$ for a word in a sentence. As such, words that occur frequently in a sentence yet are relatively rare across the entire task will weigh more, indicating their importance for the meaning of a sentence. TF-IDF weighted embeddings have shown to perform well on semantic textual similarity tasks (Arora et al., 2016). We evaluate the models using both averaging and TF-IDF weighed averaging.

5 Results and Analysis

Table 1 shows the results. It is important to note that it was not possible to perform a statistical analysis. As such, results that are very close to each other (within 0.5 points) are not considered to be significantly better/worse (compared across methods, not average/TF-IDF within the same method). For the rest, it can be observed that sentence embeddings inferred with EmbedAlign outperform SkipGram's on most tasks, only performing slightly worse on SUBJ. This indicates that overall, the multilingual context encodes useful information into word embeddings.

EmbedAlign performed better on SST, STS14, MRPC, TREC and SICK-E compared to SkipGram. The other tasks (MR, CR, SUBJ and MPQA) are all binary classification tasks, whereas these tasks have multiple classes. Perhaps, such *simpler* tasks cannot benefit from the rich context encoded with EmbedAlign.

Furthermore, the result of EmbedAlign on STS14 might be explained by the fact that different words in one language might translate to the same word in another, when they are similar. Thus, translation gives information about similarity. Similarly, classifying entailments is helped by knowledge of hypernyms and hyponyms (For e.g., "the *person* is riding the bicycle" entails "the *man* is riding the bicycle"). Again, hypernyms in one language might translate to the same word in another, perhaps explaining EmbedAlign's good result on SICK-E.

It can also be seen that the more complex TF-IDF weighting does not necessarily improve the quality of the embeddings. For both SkipGram and EmbedAlign it only led to improved results on two tasks. This may indicate two things: either TF-IDF is ineffective or the task corpora are too small TF-IDF to be effective. However, TF-IDF does show an increase in performance over averaging on STS14, which is in line with (Arora et al., 2016).

6 Conclusion

It can be concluded that EmbedAlign creates more informative sentence embeddings than SkipGram, overall. Furthermore, unweighted averaging of word embeddings produces good sentence embeddings, forgoing the need for more complicated weighted averaging methods.

However, the comparison between the models has been performed on the rather small Europarl corpus. It has been shown that word embeddings increase in quality given more data (Mikolov et al., 2013a). Therefore, future work needs to investigate how EmbedAlign would perform given a larger corpus, now it has shown to outperform SkipGram on a small dataset.

Furthermore, models that embed directly into sentences have not been investigated in this work. It would be interesting to see how those models stack up against our averaged word embeddings. Also, it would be insightful to research whether EmbedAlign could be extended to produce sentence embeddings directly, akin to Doc2Vec (Le and Mikolov, 2014).

References

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

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Aleksandar Botev, Bowen Zheng, and David Barber. 2017. Complementary sum sampling for likelihood approximation in large scale classification. In *Artificial Intelligence and Statistics*, pages 1030–1038.

Arthur Bražinskas, Serhii Havrylov, and Ivan Titov. 2017. Embedding words as distributions with a bayesian skip-gram model. *arXiv preprint arXiv:1711.11027*.

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

J.R. Firth. 1957. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

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

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.