Evaluating Sentence Representation

Tarun Krishna
University of Amsterdam
tarun.krishna@student.uva.nl

Dhruba Pujary
University of Amsterdam
dhruba.pujary@student.uva.nl

1 Introduction

Distributed representations of words induced from large unlabeled text collections had a large impact on many natural language processing applications, providing an effective and simple way of dealing with data sparsity. The word representations computed using neural networks or more precisely *Skipgram* (Mikolov et al., 2013) models are very interesting because the learned vectors explicitly encode many linguistic regularities and patterns. On the other hand, there is a different class of probabilistic models, which encodes these representations as probability densities. Intuitively, these densities represents the distributions over possible meanings of a word. Two such generative models are [1]*bayesian-skipgram* (Brazinskas et al., 2017) and *EmbedAlign* (Rios et al., 2018). However, in this report we did a comparative study of 2 models, namely *Skipgram* and *EmbedAlign* and report their performances on the quality of universal sentence representations using *SentEval framework* (Conneau and Kiela, 2018).

In next section, we briefly describe the intuitions behind word embeddings, recent advancements, and existing techniques. In subsequent sections, we describe implementation details of these models, followed by evaluation task, analysis of our results and conclusion.

2 Related Work

In the past decade, there has been an explosion of interest in word vector representations. `word2vec`, arguably the most popular word embedding, uses a continuous bag of words and *Skipgram* models, in conjunction with negative sampling for efficient conditional probability estimation (Mikolov et al., 2013). Other popular approaches use feed-forward (Bengio et al., 2003) and recurrent neural network language models (Mikolov et al., 2010; Collobert and Weston, 2008) to predict missing words in sentences, producing hidden layers that can act as word embeddings that encode semantic information. They employ conditional probability estimation techniques, including hierarchical softmax (Mikolov et al., 2011) and noise contrastive estimation. (Levy and Goldberg, 2014a) propose to use syntactic context derived from dependency parses. They show that their representations are much more discriminative of syntactic function than models based on the immediate neighborhood as in (Mikolov et al., 2013).

A different approach to learning word embeddings is through factorization of word co-occurrence matrices such as `GloVe` embeddings (Pennington et al., 2014). The matrix factorization approach has been shown to have an implicit connection with *Skipgram* and negative sampling (Levy and Goldberg, 2014b). Bayesian matrix factorization where row and columns are modeled as Gaussians has been explored in (Salakhutdinov and Mnih, 2008) and provides a different probabilistic perspective of word embeddings.

Probabilistic word embeddings have recently begun to be explored, and have so far shown great promise. (Vilnis and McCallum, 2014) They represent words as Gaussian distributions and directly

[1]SG:skip-gram, BSG: Bayesian Skipgram, EA: Embed Align

optimize an objective expressed in terms of divergences (e.g., the Kullback-Liebler divergence) between the distributions. In this paper we report evaluations of one such Probabilistic models *EmbedAlign* (Rios et al., 2018) with *Skipgram*.

3 Methodology

In this section, we give a brief description of the 2 models and their implementation details . We also provide critical practical details for numerical stability, hyperparameters, and initialization if any.

3.1 SkipGram

(Mikolov et al., 2013) This model intuitively tries to fit a binary classifier to detect whether or not a target word is likely to co-occur with neighboring words. If the binary classifier represents a word as a continuous vector, that vector will be trained to be discriminative of the contexts it co-occurs with, and thus words in similar contexts will have similar representations.

We used *gensim* for implementation of *Skipgram*. We trained different models with varying parameters. For comparison with *EmbedAlign* we considered the one trained with 30 epochs, embedding dimension 100, $negative\_samples$ 10 and a context $window\_size$ of 5. Rest of the models were used to study the effect of different parameters on *SentEval* tasks. **Training data** We used `Europarl` (Koehn, 2005) English corpus for training purposes. We employed very minimal preprocessing, namely, tokenization, lower casing and removing words with frequency less than 5.

3.2 EmbedAlign

EmbedAlign (Rios et al., 2018) model embeds words in their complete observed context and learns by marginalizing over latent alignments. It embeds words as posterior probability densities, rather than point estimates, which allows us to compare words in context using a measure of overlap between distributions. *EmbedAlign* models a distribution over pairs of sentences expressed in two languages, namely, a language L1 (language of our interest), and an auxiliary language L2 which serves as a supervision from which the model exploits some learning signal. The model is parameterized by neural networks and parameters are estimated to maximize a lower bound on log-likelihood of joint observations. The models follows a simple generative story which can be [2]summarized as:

1. sample a length m for a sentence in $L1$ and a length n for a sentence in $L2$;
2. generate sequence $z_1, ..., z_m$ of d-dimensional random embeddings by sampling independently from a standard Gaussian prior;
3. generate a word observation x_i in the vocabulary of $L1$ conditioned on the random embedding z_i;
4. generate a sequence $a_i, ..., a_n$ of n random alignments each maps from a position a_j in x_1^m to a position j in the $L2$ sentence;
5. finally, generate an observation y_j in the vocabulary of $L2$ conditioned on the random embedding z_{a_j} that stands for x_{a_j}

Entire model can be represented with a plate notation as in Figure1. The network architecture consists of a generative model which is realized using two neural networks each mapping from latent embedding to categorical distribution over the vocabulary of $L1$ and $L2$. And an inference model which is realized using two neural network mapping encoded input words to approximate posterior parameters. Encoding of input words from $L1$ is done using BOW encoder or BiLSTM. [3]**Training Data** Model was trained on bilingual data. `Europarl` with English-French Language pairs.

4 Evaluation Task

For evalution we used *SentEval* (Conneau and Kiela, 2018) framework, a toolkit for evaluating the quality of universal sentence representations. SentEval encompasses a variety of tasks, including binary and multi-class classification, natural language inference and sentence similarity. We assess quality of our representations using their framework on the following benchmarks evaluated on classification accuracy:

1. **MR** classification of positive or negative movie,reviews;

[2]taken from (Rios et al., 2018)

[3]Assuming model provided to us has same training parameters as mentioned in the paper

2. **SST** fined-grained labelling of movie reviews from the Stanford sentiment treebank;
3. **TREC** classification of questions into k-classes
4. **CR** classification of positive or negative product reviews;
5. **SUBJ** classification of a sentence into subjective or objective;
6. **MPQA** classification of opinion polarity;
7. **SICK-Entailment** textual entailment classification;
8. **MRPC** paraphrase identification in the Microsoft paraphrase corpus;

we also used correlation metric as well **STS-14** semantic textual similarity (Pearson/Spearman).

Apart from that we also used series of *probing tasks* (Conneau et al., 2018) provided by *SentEval framework* to evaluate what linguistic properties are encoded in our sentence embeddings. These task are broadly divided into :

1. **Surface Information** : These tasks test the extent to which sentence embeddings are preserving surface properties of the sentences they encode. This includes **SentLen** and *word content* **WC** task.
2. **Syntactic information**: These batch of tasks test whether sentence embeddings are sensitive to syntactic properties of the sentences they encode. This includes *bigram shift* **BShift**, *tree depth* **TreeDepth** and *top constituent* **TopConst** tasks.
3. **Semantic information** These tasks also rely on syntactic structure, but they further require some understanding of what a sentence denotes. These includes series of tasks **Tense**, *subject number* **SubjNum**, *object number* **ObjNum**, *semantic odd man out* **SOMO** and *coordination inversion* **CoordInv**.

The classifier in the *SentEval* framework is trained with the default settings with 10 fold cross-validation.

5 Result and Analysis

The sentence embeddings of *Skipgram* and *EmbedAlign* models are tested on different NLP task(downstream and probing) using the toolkit *SentEval*. Sentence embeddings are obtained by simply averaging over all word embeddings of each word in a sentence for both the models. From Table1 we see that the *Skipgram* performs better than *EmbedAlign* in the task such as MR, SST-2 and CR. *Skipgram* could perform better in the task of sentiment analysis because the word embedding could have captured the concrete meaning of positivity or negativity from the words within context window which is reflected in the sentence embedding as well. Whereas *EmbedAlign* which captures a more general meaning of the word could not distinctly separate the sentiment in the sentence upon averaging. *Skip-gram* also performed better in TREC, MPQA and SUBJ because of a similar explanation as above. *EmbedAlign* performs slightly better than *Skipgram* for MRPC and significantly better for the SICK-Entailment task. This could be explained by property of *EmbedAlign* that it captures the meaning based on words occurring in the sentence more generally or more technically it encodes a density of meaning for the word.

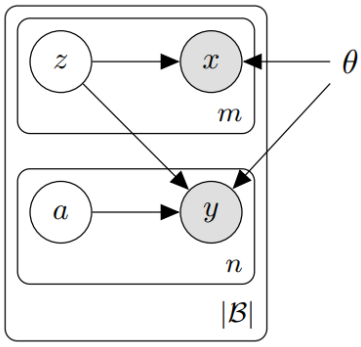

Figure 1: Plate notation for EmbedAlign

From Table2 we see that for the semantic textual similarity task (STS-14) the *Skipgram* performs better in the different test sets and overall in general. Also, note that difference between the positive correlation is slightly more for Pearson-correlation than Spearman-correlation in the two model.

The sentence embeddings are also tested on probing tasks included in the toolkit. From Table3 we see that *EmbedAlign* is better in predicting accurately the length of the sentence (SentLen) from the sentence embedding than *Skipgram*. This is because of the embedding generation process of *EmbedAlign*

Model	TREC	SST2	MR	MPQA	SUBJ	CR	SICK-E	MRPC
Skipgram	59.60	69.58	68.1	84.75	82.52	72.72	67.28	70.84
EmbedAlign	57.4	67.05	64.72	83.78	79.15	70.65	74.75	70.96

Table 1: Accuracy of Skipgram and EmbedAlign on different NLP (downstream) Task (in %)

Model	**deft-forum**	**deft-news**	**images**	**OnWN**	**headlines**	**tweet-news**	**all**
Skipgram	0.39/0.37	0.69/0.66	0.73/0.70	0.76/0.78	0.58/0.55	0.68/0.62	0.65/0.63
EmbedAlign	0.35/0.36	0.62/0.58	0.66/0.64	0.66/0.72	0.58/0.58	0.62/0.55	0.59/0.59

Table 2: **STS-14** Pearson/Spearman of Skipgram and EmbedAlign sentence embedding on Test set(in %)

Model	**WC**	**Tree Depth**	**SentLen**	**TopConst**	**BShift**	**SOMO**	**Tense**	**Coord Inv**	**Subj Num**	**Obj Num**
Skipgram	65.18	22.41	21.87	15.94	49.76	50.16	66.72	50.34	66.2	69.22
EmbedAlign	38.40	23.74	34.54	32.64	51.13	50.16	67.46	51.05	70.78	68.45

Table 3: Accuracy of Skipgram and EmbedAlign on different NLP (probing) Task (in %)

which takes into account the whole sentence rather than limiting within a context window. It also performs slightly better for TreeDepth than *Skipgram*. *Skipgram* performs significantly better than *EmbedAlign* in WC task because the *Skipgram* embedding absorbs the words appearing within the context window in its word embedding and transfer these properties in the sentence embedding as well.

The accuracy of BShift task of *EmbedAlign* is slightly better than *Skipgram* because *EmbedAlign* has more sense of word order(uses BiLSTM for word representation). For TopConst task, *EmbedAlign* performs significantly better than *Skipgram*, and for Tense and Subj Num task, the *EmbedAlign* model performs better than *Skipgram*. Whereas for the task of Obj Num it is otherwise. The better performance of *EmbedAlign* on these tasks can be attributed to the fact that the word embeddings has more sense of the meaning of the whole sentence rather than few words in its surrounding. Similarly for CoordInv *EmbedAlign* performs better.

We also tried to study the effect embedding dimension for *SkipGram* for NLP (downstream) Task. We trained different sets of model with varying embedding dimensions precisely 100, 150, 200 and 300 while keeping other parameters constant i.e $window\_size = 5$, $training_epochs = 20$,$negative\_samples = 10$. Figure2 clearly shows how changing embedding dimension significantly changes improves the accuracy with the only exception in MRPC task.

6 Conclusion

In this report we have analyzed two types of sentence embedding namely *Skipgram* and *EmbedAlign* on different NLP task using a toolkit *SentEval*. In most of the downstream task, *Skipgram* performs better than *EmbedAlign*. Whereas for most of the probing task *EmbedAlign* perform better than *Skipgram*. The results obtain does show that one embedding is better than the other depending on the task. However, these variations could also be because of difference in training settings of the two models. To have a concrete answer, future work would be to test and analyze different training settings of both the models and perform significance tests.

In addition, we tested *Skipgram* for different embedding dimension which shows that increasing the embedding dimension improves the accuracy of downstream tasks except for MRPC. It is worth to notice that *EmbedAlign* performed better than *Skipgram* in this specific task. This can be analyzed as part of the future work.

Code:
https://github.com/druv022/ULL/tree/master/Lab3

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic lan-

guage model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Arthur Brazinskas, Serhii Havrylov, and Ivan Titov. 2017. Embedding words as distributions with a bayesian skip-gram model. *CoRR*, abs/1711.11027.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *CoRR*, abs/1803.05449.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties.

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.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In Roser Morante and Wen tau Yih, editors, *CoNLL*, pages 171–180. ACL.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Tomas Mikolov, Martin Karafit, Luks Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

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. *CoRR*, abs/1802.05883.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR*, abs/1412.6623.

A Skipgram parameters

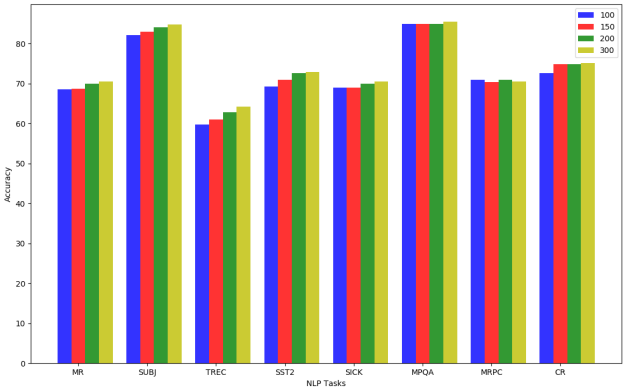

Figure 2: Effect of embedding dimension on downstream tasks.