Unsupervised Language Learning: Lab 2

Aron Hammond & Joris Mollinga

10437215 & 11871431

Code can be found on `https://github.com/jorism1993/ULL-Practicals`

Skip-Gram Model

We implemented the Skip Gram model described by [Goldberg and Levy2014] using negative sampling and batches. We trained the model on the 18,000 most common words to reduce computational overhead. We chose an embedding dimension of 300 and trained the model for 100 epochs using a batch size of 50. This took us approximately 12 hours on a CPU. After 100 epochs the loss had not yet stablized, indicating room for improvement. This model starts to capture word similarities, for example the closest 10 words of the word *money* include *funds* and *laundering*, but there are also quiete a few irrelevant words. We implemented the evaluation method used by [Melamud et al.2015], using a combination of both word embeddings and context embeddings. This method should give better results than only using the target or context embeddings because it takes the context of the surrounding words into account. To determine a target's word context we use a window size of 1. This provided us with the best results.

Bayesian Skip-Gram Model

The implementation of the Baysian Skip-gram model (BSG) is directly based on [Bražinskas et al.2017]. Our implementation seems to work on the `dev_en` set. But when trying to fit on the training set (with any hyper parameter setting) the model doens't learn anyting. The GAP when training on the 10.000 most frequent words and an embedding size of 100 is 0.293. Comparing this to random initialization (0.31) and training on `dev_en` (0.30) it seems that training actually hurts the model's performance on the lexical substitution task.
A possible explanation comes from the observation that training the model pushes the weights of the internal embeddings to negative values, resulting in very sparse vectors after applying the ReLU. The cause for this is subject to further analysis. A solution might be to experiment with different activation functions.

Embed Align Model

We implemented the Embed Align Model presented by [Rios et al.2018]. In an attempt to reduce computation times we limited all the multi-layer perceptrons in the proposed architecture to one layer, instead of two. However, computation times where still very high, approximately 8 hours per epoch on a CPU. Unfortunately while training the Embed Align Model the python kernel crashes, resulting in an untrained model. Because of the deadline of this report, we were unable to try it again. Results of the Embed Align Model can therefore not be subjected to a qualitative analysis.

Evaluation

We implemented the lexical substitution task where we score candidate solutions using a set of gold standard rankings and report performance in terms of Generalized Average Precision (GAP). We do not take multi word embeddings into account by setting the `no-mwe` option. Results are summarized in Table 1.

Table 1: Performance score of the various models

	Skip Gram	Bayesian Skip Gram	Embed Align
Performance (GAP)	0.31	0.29	0.30

The Skip Gram Model gave us the best performance on the lexical substitution task, which is contradictory to literature. One would expect better performance of the Bayesian Skip Gram and Embed Align model because these models are designed to take context into account [Bražinskas et al.2017][Rios et al.2018]. Because of the sparse vectors in the Bayesian Skip Gram Model and the crash of the Embed Align Model these results are very poor. The alignment error rate of the Embed Align model is 0.89. Because our model crashes during training, we can not draw a conclusion on the performance of this model.

Conclusion

Implementing the models proved to be a challenging task which we executed with mixed rusults. For all three models we can conclude that we lack the computational resources needed to do a proper evaluation. We recommend running our code on a powerfull workstation to evaluate the performance of our models.

References

[Bražinskas et al.2017] Bražinskas, A., Havrylov, S., and Titov, I. (2017). Embedding words as distributions with a bayesian skip-gram model. *arXiv preprint arXiv:1711.11027*.

[Goldberg and Levy2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

[Melamud et al.2015] Melamud, O., Levy, O., and Dagan, I. (2015). A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7.

[Rios et al.2018] Rios, M., Aziz, W., and Sima'an, K. (2018). Deep generative model for joint alignment and word representation. *arXiv preprint arXiv:1802.05883*.