Why does everyone use the dot product?

Anonymous ACL submission

Abstract

This paper investigates other measures of word embedding similarity/distance by training embeddings using the skip-gram with negative sampling method and then evaluating their performance on a sequence classification task using a recurrent network. The evaluated distance measures are the dot product, cosine similarity, and vector norm distance using various p-norms, as well as each of these measures but with embedding vectors clipped to a maximum norm dependent on the dimension of the vector. The code is available.[1]

1 Introduction

Researchers of neural networks seem to strongly favor the use of the dot product to assess the similarity (or proximity or distance) of vectors. For example, the "skip-gram with negative sampling" algorithm for generating word/token embeddings was designed using a dot product to score similarity of vectors (Mikolov et al., 2013a,b), and the Transformer architecture uses a scaled dot product (Vaswani et al., 2023).

However, the reason for preferring the dot product is unclear to me. Is it due to computational efficiency? Is it due to performance? Is it just historical, since it's what's always been done? Seeing as word embeddings are the foundation for so much in the intersection of neural networks and natural language processing, it seems important to investigate whether there are better similarity measures out there, the fruits of which investigation could benefit larger efforts and models. This paper mainly addresses the performance of these similarity measures, but additional research into temporal complexity would be in order (and insight into the history of this topic would be interesting too).

[1] https://github.com/Burt360/word-embeddings

2 Related Work

It seems there is an understanding that there are other alternative approaches to measuring similarity between embedding vectors (see, for example, this Google Developers article[2]). However, I am not aware of research dedicated specifically to comparing them. Moreover, vector p-norms besides the standard 2-norm (Euclidean distance) don't seem to make an appearance at all. As this paper will demonstrate, the 3-norm yields comparable if not better results than the 2-norm, rendering this omission from the research not insignificant.

That said, some research has experimented with vector similarities using cosine similarity. (Terragni et al., 2021) proposes several similarity measures for collections of embedding vectors (such as centroid-based or pairwise approaches), pertinent to the field of describing document topics. Additionally, (Li and Summers-Stay, 2020) investigates using embeddings in a dual-space of semantic and functional meaning, and evaluates various cosine similarity-based measures of words by either concatenating the semantic and functional embeddings, or multiplying the semantic and functional similarities to obtain an overall similarity.

3 Methods

3.1 Training embeddings

For training the embeddings (as well as evaluating them, discussed in section 3.2), I use the Brown corpus (Francis and Kučera, 1964). I prepare whitespace-separated tokens, leaving tokens as-is, without normalizing into all-lowercase or similar, and I replace tokens with fewer than 5 occurrences in the entire corpus with a special <UNK> token.

Following the skip-gram with negative sampling

[2] https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity

algorithm (Mikolov et al., 2013a,b), I prepare true contexts and negative samples using a window size of 5 tokens and $k = 2$ negative samples for every true context. When sampling negative contexts, I weight frequencies by exponentiating them to $\alpha = 0.75$, increasing the probability of sampling rarer noise words (see Jurafsky and Martin, pp. 122–123).

I then train the embeddings for each of the following similarity measures:

- dot product,
- cosine similarity, and
- vector norm distance for $p = 1, 2, 3$.

I keep everything else the same between runs, including fixing a random seed so that each model type sees the same data in the same order. Note that a high dot product or cosine similarity indicate "close" vectors, while a high norm distance indicates vectors that are far apart. As such, I negate the norm distances so that the model's objective is to maximize the "similarity" scores, whatever the measure. I save each measure's resulting embedding for performance evaluation later (see section 3.2).

For each of the similarity measures I also train a second set of embeddings that clips any embedding vector that is above a threshold. To derive this threshold, I first note that the embeddings are initialized from a standard normal distribution. It turns out that the expected norm of such a vector increases with some dependence on the dimension of the vector (as well as the p-norm used). Thus it would likely not make sense to use the unit ball as the clipping boundary, since this would correspond to vectors in larger dimensions coming from normal distributions with smaller variances, which seemed an undesirable quality. (However, future research could empirically investigate the effects of such an alternative method, as well as the mathematical relationship.)

To hopefully counteract this behavior, I take a sample of these embedding vectors in the given dimension, take the mean norm with the standard Euclidean distance (unless the similarity measure is a p-norm with $p \neq 2$, in that case using that norm instead), and then set the clipping threshold at the somewhat arbitrary value of two times this mean norm.

Having done all this, I now have 10 sets of embeddings for evaluation.

3.2 Evaluating embeddings

I evaluate each of the embeddings through a sequence classification task. The Brown corpus classifies every document with one of 15 classes (such as 'adventure' or 'science'). I first prepare the corpus into a dataset of paragraphs. However, some classes have as few as 254 paragraphs. Since the purpose of this investigation is to evaluate the performance of the embeddings rather than develop a strong document classifier, I drop any class having fewer than 1000 samples, leaving 10 classes.

The model is simple: given any of the sets of embeddings obtained previously, a paragraph is embedded, then run through a multilayer recurrent neural network composed of gated recurrent networks (or GRUs, as in Cho et al., 2014), followed by a linear layer onto the output classes.

4 Evaluation

Included in this paper are four figures: one comparing the losses when training embeddings; one the validation losses when training the sequence classifier using each embedding; and two the sequence classification scores these embeddings achieve on a test set (micro- and macro-precision, -recall, and -F1-score, along with accuracy). In all cases it seems clear that the dot product performs the best and achieves the lowest losses. The 1-norm similarly performs worst. However, the other three similarity measures do not maintain a consistent ranking.

Figure 4 draws from the same data as Figure 3 but focuses on the dot product and cosine similarity scores for clipped- versus non-clipped-norm embeddings. Surprisingly, the clipped varieties beat out the non-clipped varieties on a test set in all but one score. This is especially interesting since cosine similarity effectively normalizes embeddings when computing similarities, though not when using the resulting embeddings, which may have larger or smaller norm.

Additionally, during training of the embeddings for each model, at the end of each epoch I output the six tokens that achieve the highest similarity to a few handpicked tokens (`dog`, `milk`, `run`, `apple`, and `hurt`, chosen to include some verbs and some nouns but otherwise arbitrary). I did this mainly to ensure that the embeddings were in fact gaining meaning as the training progressed. However, these outputs may provide some insight into the quality of the embeddings for each

similarity measure. Due to time, this analysis is omitted, but the interested reader may refer to `train-embeddings.ipynb` file in the repository linked above.

5 Conclusions

These preliminary results, though far from conclusive, suggest that there is reason to favor the dot product over the other measures of similarity. Likewise, the 1-norm universally yielded the worst results. However, cosine similarity and the 2- and 3-norms yielded similar results (hey, a pun finally). Moreover, the 3-norm performed somewhat better than the 2-norm in all analyzed statistics, suggesting that there is more research to be done in this area. Lastly, the clipped-norm embeddings scored a little better than the non-clipped versions during testing of the sequence classification models.

5.1 Future Work

Due to time constraints and scope, this investigation only scratched the surface for training and evaluating different similarity measures. For one, this paper only evaluated embeddings on one task, namely sequence classification. Further work could investigate many more tasks, such as part-of-speech tagging. It would be very interesting if some similarity measures performed better or worse depending on the task.

Moreover, one could research the effect of using weighted linear combinations of multiple distance measures, and even letting the model learn this weighting. Additionally, it's conceivable that a model using a p-norm distance could also learn which p to use altogether.

Additionally, it would be interesting to investigate normalizing embeddings onto a ball of some radius, rather than just clipping embedding vectors whose norms grow too large as investigated in this work. As mentioned, the clipped-norm embeddings performed marginally better than the non-clipped versions, suggesting additional research in this area may be fruitful.

Perhaps more interesting to the mathematician than the linguist, I was unable to find a mathematical relationship between the expected p-norm of a vector drawn from a standard normal distribution of arbitrary dimension and arbitrary p. (But for those interested, the repository for this work contains an `experiment.ipynb` notebook with preliminary numerical investigation into this relationship.)

But perhaps the most glaring deficiency in this work is the minimal amount of training and hyperparameter tuning. Future work could try different window sizes and numbers of negative samples per true context when training embeddings, and could train the two models described in this paper for more epochs and more data. I also only tried the 1-, 2-, and 3-norms, but there are other p-norms to try. It's possible, for example, that a e-norm or a π-norm hits a sweet spot, with lesser and greater values of p yielding diminishing performance. (Wouldn't that be fascinating?)

In addition, when designing the sequence classification model for evaluating the embeddings, I used the embeddings derived from cosine similarity to find hyperparameters that would yield a model that actually trained, at least in that case. However, more robust future work could perform a grid search on hyperparameters for each set of embeddings separately, to avoid biasing the results toward one similarity measure's embeddings.

Finally, an aim of this paper was to analyze the temporal complexity of each of the similarity measures. Due to time, however, this goal too has been laid to rest in this section.

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

W. N. Francis and H. Kučera. 1964. Manual of information to accompany a standard corpus of present-day edited american english, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island. Revised 1971. Revised and amplified 1979.

Dan Jurafsky and James H. Martin. *Speech and Language Processing*.

Dandan Li and Douglas Summers-Stay. 2020. Dual embeddings and metrics for word and relational similarity. *Annals of Mathematics and Artificial Intelligence*, 88(5):533–547.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781 [cs].

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. ArXiv:1310.4546 [cs, stat].

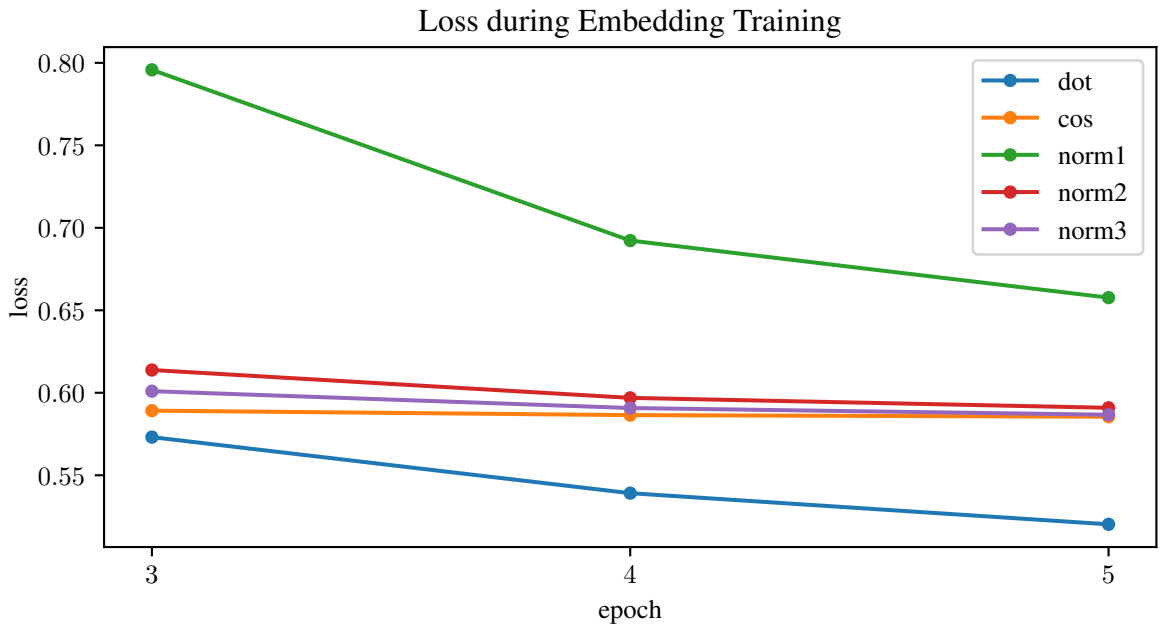

Figure 1: The loss during the training of the embeddings with the skip-gram with negative sampling algorithm. Note that the loss function and data are identical for each similarity measure. The dot product is the clear winner, with the 1-norm in last and the others relatively close. The plots for the clipped-norm variants are similar. (Epochs 1 and 2 are omitted to focus and zoom in on the loss at the later epochs.)

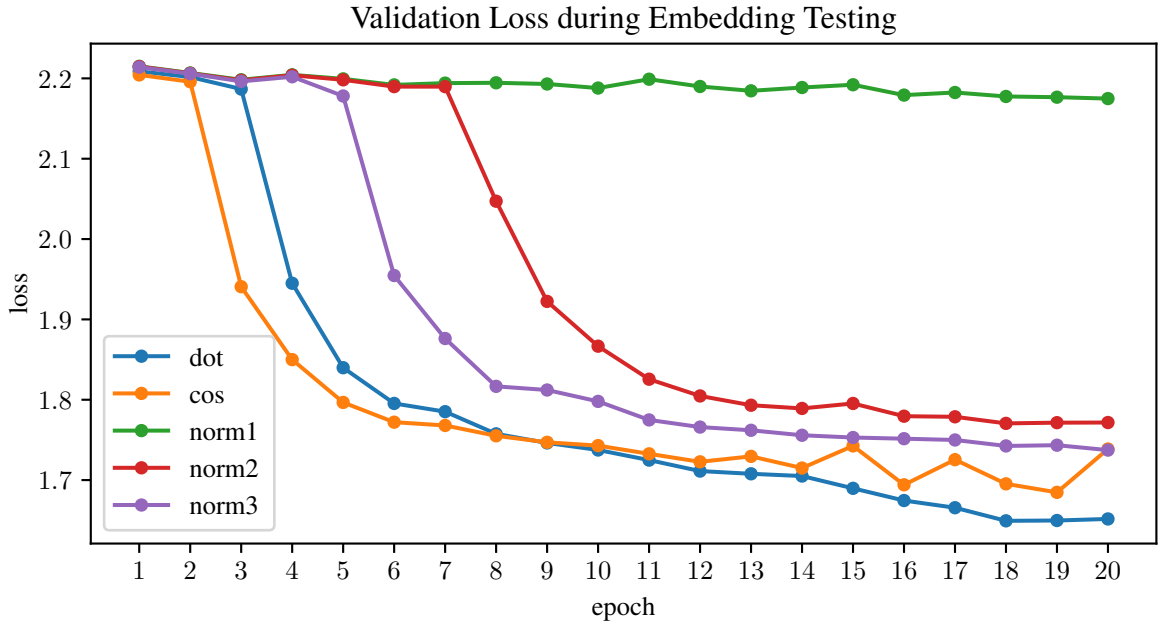

Figure 2: The validation loss during the training of the document classifier. Again, the dot product is in the lead, with the 1-norm in last and the others in between. Interestingly, the 3-norm beats achieves lower loss than the 2-norm for this number of epochs. The plots for the clipped-norm variants are similar.

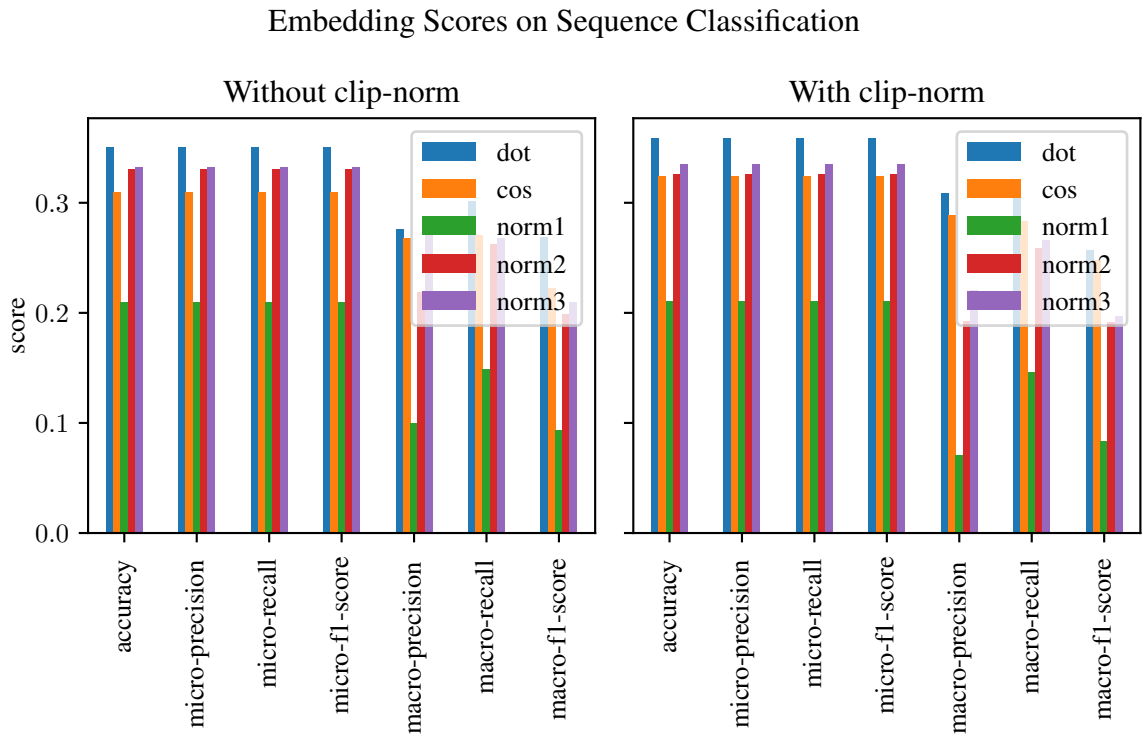

Figure 3: Scores for the document classification model on a test set using each set of embeddings. The rankings of the embeddings are similar to those suggested by the losses (seen in Figures 1 and 2).

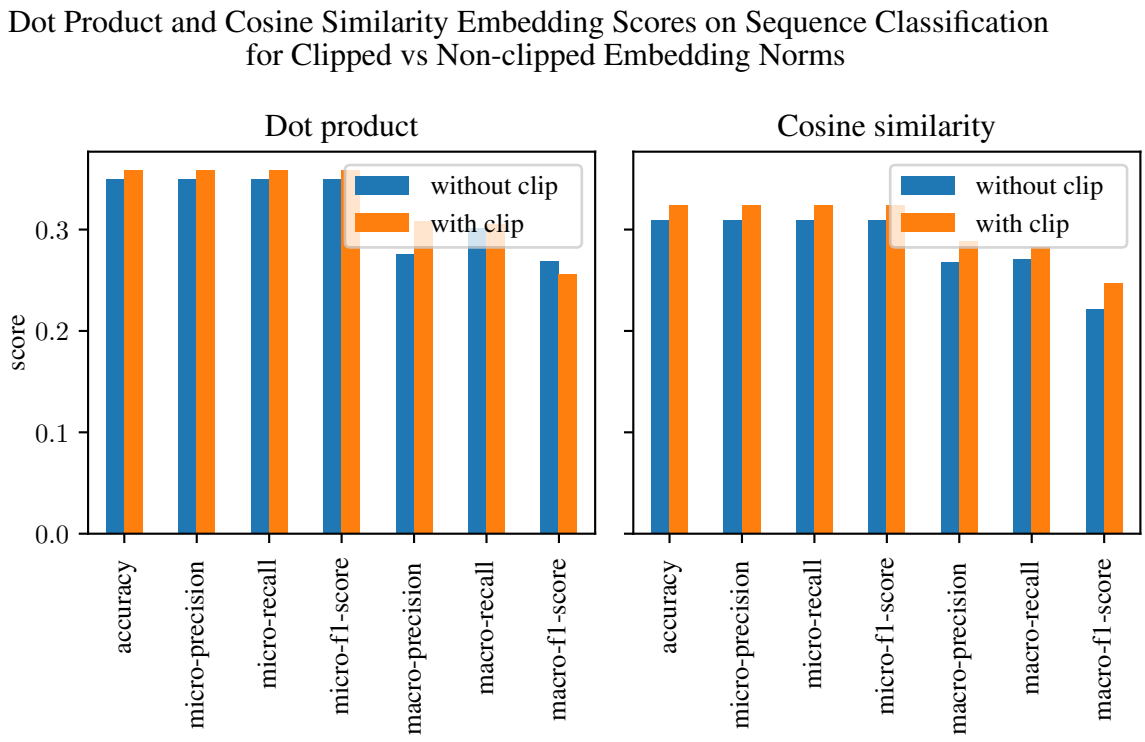

Figure 4: This figure is taken from the same data as Figure 3 but emphasizes the difference between the dot product and cosine similarity, and between clipping and not clipping norms. Again, the dot product leads out. Notably, the clipped-norm embeddings beat the corresponding non-clipped version in all but one case among the score comparisons pictured here.

Silvia Terragni, Elisabetta Fersini, and Enza Messina. 2021. Word Embedding-Based Topic Similarity Measures. In *Natural Language Processing and Information Systems*, pages 33–45, Cham. Springer International Publishing.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. ArXiv:1706.03762 [cs].