LING 575 —Text Representation —Term Paper

Peter Schoener

Abstract

In this project, we attempt to create sentence embeddings from word embeddings by use of a recursive but otherwise shallow neural network.

1 Motivation

Distributional word embeddings have been established as an effective way of representing word semantics and, to some degree, syntax. More recently, similar methods and, more broadly, vectorization techniques, have been applied to complete sentences with some success. However, the methods by which the sentence embeddings are derived are generally not based on the syntax of the sentence, which is instead approximated by a recurrent or convolutional model that takes only surface-level ordering into account.

By considering the structure of a sentence, it should be possible to better model the interactions between words, thus creating a better sentence embedding. While the target in this instance is a sentence embedding generated with a different method, this type of model should still better approximate the specific relations between the words. This means, among other things, that it should at least generalize better to strange sentence structures.

Moreover, having a specific transformation associated with each word and relation means that linguistic analysis of the transformations for theoretical purposes should be easier. The ways in which these transformations generalize (or fail to do so) may be of use, for example in engineering tagsets for parser grammars.

While it may be possible to certain short phrases the same way as words, the amount of data needed to find the same number of instances increases exponentially with the length of the phrase. This means that in order to generate sentence embeddings with non-compositional techniques, the corpus and computing resources needed would quickly become impractical. Meanwhile, embeddings for phrases or sentence fragments would be very useful for processing or evaluating informal language, such as that found on social media.

2 Related Work

Since their introduction, in which they were already shown through relatedness and analogy tests to model meaningful semantic properties (Mikolov et al., 2013), embeddings have been applied with great success to a variety of semantic tasks. These include everything from sentiment analysis (Tang et al., 2014) to document classification (Kutuzov et al., 2016) to machine translation (Mi et al., 2016). It is clear, then, that they are a sound technique for representing semantics.

Of course, many applications of word embeddings would be better served by embeddings of the complete sentences. Therefore, approaches have arisen with the goal of creating the same sort of vector representations for complete sentences, some with analogous methods to word embeddings. One early success was with the skipthoughts model (Kiros et al., 2015), which, much like word2vec, involved training a model to predict context. However, this model was based on a recurrent encoder/decoder, which considered only the order of words, not their linguistic relations to each other. Other models, such as BERT (Devlin et al., 2018), have created incredibly powerful representations, but these are still based on sequence rather than structure, and are therefore neither introspectable nor guaranteed to generalize.

There is clearly an argument in favor of using word embeddings in a recursive way according to

sentence structure. Even before many of these sentence embedding approaches became available, a recursive neural net was used to great effect in sentiment analysis with what were essentially very compact vector representations of words and phrases which propagated upward through a parse tree to give a final result (Socher et al., 2013). While these polarity embeddings were not as complex or high-dimensional as most word embeddings, the success of this approach underlined the weaknesses of ordering-based approaches to sentence meaning; many subtleties of structure, which can prove vitally important to the correct understanding of a sentence, are lost when its internal structure is not directly considered.

There is considerable work on composing word embeddings in order to create phrase embeddings (Dima, 2015; Tai et al., 2015; de Kok et al., 2017), which have the variety of applications described above, but many of these focus (for now) on composing only a few words in order to arrive at the embedding for a compound made up of them. While this would be highly useful, it also has its limitations, for example not necessarily accounting for all relation types, but rather only those that would exist between the leaves of a parse, e.g. the relations between the elements of compound words.

One common goal of embedding composition methods is that the composed embeddings be of fixed length. This has obvious advantages: many of the ways in which embeddings are used rely on operations, for example addition for analogy or cosine for similarity, which are only defined over operands of equal length. Therefore, it would be useful for any two sentences generated by an embedding method to be compatible with each other in this way. This rules out the trivial method of simply concatenating the embeddings.

There has also been work (Schoener, 2018) on checking semantic compatibility as a prerequisite for embedding composition, in which successful composition methods were applied to compatibility checking for, among other things, parsing.

3 Approach

As described above, the goal of this project is to model sentences in vector space by recursively composing them from the word level using pretrained word embeddings and learned transformations which correspond to the relations in a parse. However, it is not without reason that such a theoretically justifiable approach has not been extensively tested.

Recursive models are very difficult to train, since the size and structure of the computation graph will vary wildly between each training sample. This means that the simplest approach is to train based on each sample separately rather than in batches, the advantage of which would be generalizability. The advantages of batching would be especially useful in this case because the complexity of the networks means that it wuld be easy, without batching, to train the wrong part of the model.

Another challenge is the computational power needed to backpropagate through what is essentially a very deep network. Much like a recurrent neural network, the training procedure involves essentially unrolling the model and training several layers at once, only in this case with a tree structure instead of a simpler feed-forward structure.

The third main challenge in training this network is not directly related to the nature of recursive neural networks, but is related to the nature of sentence embeddings. Because we are looking to keep the embeddings at a fixed length while still being able to model the information contained in a sentence, the vectors must be large. Skipthoughts in its default state uses 4800 dimensions in order to capture the meaning of a sentence, meaning that a sentence can be represented to at least the standard of skipthoughts in at most that many dimensions.

The first version of the model was designed without any batching other than what was inherent in the training of each individual sample — certain transformations can be used more than once in a sentence, meaning that they will be trained, much like in a batch, on multiple environments at a time. However, this model failed to generalize, so we expanded the model to train a complete epoch in each pass.

In order to limit the depth of the network, it was necessary to only train on short sentences. This not only makes graph construction faster, but also dramatically reduces the complexity and time needed during backpropagation. For simplicity, sentences were selected based on length, rather than actual parse depth, with the cutoff (when used) ranging from three to ten words.

Because of the high dimensionality needed to represent the full meaning of a sentence, it would

make no sense to pass the representations through a smaller hidden layer, thereby creating a bottleneck and losing information. Although still attempted, hidden layer sizes under 4000 units proved useless. The hidden unit size was, however, limited to 14,400 due to limited computing resources.

Each composition involved creating a transformation by passing the marginal word through a layer selected according to the relation type, which then yielded the transformation by which the existing embedding would be transformed. This two-layer approach was somewhat necessary in order to incorporate both the word and the specific relation into the new embedding. Of course, it would also have been possible to concatenate the two and pass them through a general layer, but this would have lost the specificity of a dependency-type-specific transformation. However, the depth of this network was already bordering on problematic, even with sentences of average length.

After each layer a cut off sigmoid is applied, allowing the model to preserve the signedness of individual dimensions (unlike with a ReLU) while also normalizing the vector's length, giving it desirable properties in terms of comparability to other vectors. The cutoff is used to prevent the overadjustment commonly caused by the sigmoid's asymptotes.

The loss function was the cosine distance from the skipthought embedding for the sentence, which means the model does not directly relate to the semantics of the sentence and relies instead on the effectiveness of a different sentence embedding model. However, this was more or less necessary given the timeframe of the task; training a sentence embedding model just once on contexts would already have taken more than the specified duration of the project.

The inputs are drawn from the Universal Dependencies corpus. This ensures high quality, consistent annotations and what should be sufficient volume for training.

4 Results

During training, it became clear that the model would not, at least within the confines of its maximum tractable hyperparameters, be able to approximate skipthought embeddings. The vectors are initially almost perpendicular, with their cosines only declining to a test loss of 0.9994 on train and 0.9996 on test data. To achieve this optimal performance, hidden layers of 14400 units are used, with 200 epochs of training, shortly before the end of which the model's performance plateaus.

However, because of the indirectness of the training method, the usefulness of the vectors is not directly tied to the training objective. In order to do a cursory comparison of the model with skipthoughts, both were tested using a linear classifier on a small polarity judgement task. The reasoning behind using a simple linear classifier is the same as in the original skipthoughts paper: this way, the direct applicability of the embeddings can be tested, as opposed to the applicability of a technique.

Both the skipthoughts and the recursion model scored 54.7% accuracy on the polarity test, which equates exactly to a guess; the models both simply guessed positive for all samples. Clearly, either both methods are flawed or they require a larger dataset for this task.

5 Discussion

One of the alarming parts of the performance of the model was the extremely high loss. In every tested configuration, the vectors started out perpendicular to each other and only very slowly began to align. Because only the zero vector is perpendicular to everything, this suggests that for some reason the trained vectors start with a great number of zeroes, of which only few are initially trained. Better initialization or an optimizer that makes smaller adjustments to more weights might be a solution to this problem, but for neither could an example yet be found through trial and error.

The main issue, at least as far as could be determined from the results, was computational efficiency and power. The largest trainable models were still showing marginal improvements both with added sentence length and larger hidden layers. Whether a deeper network would have performed better could not be meaningfully tested due to how much smaller the layers would have needed to become in order to train the model on the available hardware.

However, although the model was still substantially improving, it may hit a plateau before it actually reaches an accuracy at which it becomes useful. Also, at this rate, the size increase required to reach that point would be impractical, meaning

that it would be wise when expanding the model to also look into increasing the power of the individual layers.

Although previous work on embedding composition shows that this is very much a nontrivial task even when only composing two embeddings, the idea here was that considering the entire sentence may yield a more useful target than the one approximated by a compound word. This does still theoretically hold water, but embedding composition is, in fact, still difficult with this approach, possibly even more so than with the shallower training methods.

Due to the optimization issues, the possibility of compressing each recursive segment to have only a single hidden layer may be worth exploring. Perhaps by limiting the model to half as many hidden layers it would become feasible to make the layers substantially larger than the sentence embeddings themselves, allowing for information to be better retained.

6 Conclusion

In this project we attempted to create a recursive compositional model for word embeddings which could generalize up to the sentence level. Although unsuccessful, the results were promising enough that the approach is worth retooling and retesting on more powerful hardware. This project will therefore continue development in the near future.

The testing will need to be rerun on a larger dataset to verify that the problem is in fact with the embeddings.

7 Bibliography

References

Daniël de Kok, Jianqiang Ma, Corina Dima, and Erhard Hinrichs. 2017. PP attachment: Where do we stand? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 311–317. https://www.aclweb.org/anthology/E17-2050.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Corina Dima. 2015. Reverse-engineering language: A study on the semantic compositionality of german compounds. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1637–1642.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Andrey Kutuzov, Elizaveta Kuzmenko, and Anna Marakasova. 2016. Exploration of register-dependent lexical semantics using word embeddings. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*. pages 26–34.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. *arXiv preprint arXiv:1605.03148* .

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

Peter Schoener. 2018. *Embedding-based approaches to prediction of semantic compatibility*. Bachelor's thesis, Eberhard Karls Universität Tübingen.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1555–1565.