Analyzing the compositional properties of word embeddings

Thijs Scheepers
University of Amsterdam
thijs.scheepers@student.uva.nl

Efstratios Gavves
University of Amsterdam
e.gavves@uva.nl

Evangelos Kanoulas
University of Amsterdam
e.kanoulas@uva.nl

ABSTRACT

We present an in-depth analysis of various word embeddings in terms of their compositionality. Measuring the quality and usefulness of single word embeddings[10] and their compositional representations is a notoriously difficult task often tackled by small tests using handpicked data. Understanding the compositional effectiveness on real data can help improve models which use word embedding composition, such as attention mechanisms [1] as well as document representations [9]. In our analysis, we evaluate using multiple methods for word embedding composition and test these methods on a dictionary based dataset. This gives us the ability to evaluate word embeddings quantitatively, as well as give direction to architecture considerations for neural networks in which these word embeddings are composed. Additionally our dataset could be used for further research into the compositionality of word embeddings.

1 INTRODUCTION

In linguistics there is a basic principle called *compositionality* [6]. The principle states that the meaning of an expression comprises the meaning of its constituents as well as the rules to combine them. This principle was conceived to explain the way humans understand language, however it could be applied to the language understanding by machines as well.

Real valued vector representations of words, i.e. *word embeddings* [10], have become a important aspect of deep neural models for natural language processing, information retrieval as well as other text related model classes. Ever since the paper which made these real valued embeddings popular there has been interest in their compositional properties [11]. These properties are especially important in the context of deep neural models where, in various architectures, multiple representations can be composed into a single deeper representation.

Finding good representations for words in an unsupervised manner often relies on a word's context. Training based on context results in a representation which captures both syntax as well as semantics. While syntactic information is important for composing representations, it is not necessarily useful for using the meaning of single word embedding in a model. When composing single words into a joint representation we often want to create a representation of meaning, i.e. semantics, and often do not care for syntax, even though this information is essential to the act of composition itself.

Having a compact representation of meaning can be useful for lots of tasks. While creating this compact representation of meaning we have to consider compositionality. For example, a popular method for creating paragraph representations is called Doc2Vec [9], in which word vectors are averaged as well as combined with a separate paragraph representation. Such a combined representation can then be used in document retrieval. This method makes the assumption that averaging is a good method for composition.

In more complex models, like models for neural machine translation [3] (NMT) or neural question answering (NQA) the word embeddings are trained jointly with the model in a supervised manner. The embedding-matrix in these models are good candidates for transfer learning from the unsupervised context-driven approach to jump start training. When applying transfer learning it is important to consider the compositional properties of the used embeddings. Additionally, in the case of these types of models, one should consider the effect its architecture has on the compositionality of representations at various points in the model. For example, the encoder in a traditional sequence-to-sequence NMT model uses semantic and syntactic information to generate a good sentence representation through an recurrent neural network (RNN). However the decoder is only interested in a representation of the semantics of the encoded sentence. But within the encoder the hidden state should still contain syntactic information to allow for the composition to happen properly for each encoding step. So inherently the model is not optimized for pure semantics at the start of decoding.

When we look at attention [1], which is also an important component of NMT and NQA systems [18], creating an attention vector boils down to using a different method for composition, as opposed to RNN encoding. In a traditional attention architecture multiple assumptions are made on how they compose representations, e.g. using the hidden states from the encoder as input and using a weighted average over all source words or a specific window.

Evaluating word representations in general is a difficult task. This usually happens in terms of the similarity between two words and is handcrafted for specific examples [5]. Downstream performance in applications such as sentiment classification provide an extrinsic evaluation of compositionality, but results may suffer from other confounding effects that affect the performance of the classifier [15]. Additionally there are analogy tasks which you could use to evaluate embeddings in terms of both semantics as well as compositionality [11, 14] instead of just only looking at semantics, however they are even more limited and specific in scope.

In this work we introduce a novel approach for the evaluation of the compositional properties of word embeddings. Our method allows for a more general evaluation of compositionality with data which is not hand picked [12], therefore results are expected to be closer to composition in real applications. To test our method we evaluate well known pre-trained word vectors and find interesting results. Hence, the contribution of this paper is two-fold: (1) we propose an novel evaluation framework for testing word embedding compositionality; (2) we use the proposed framework to test popular word embeddings and popular aggregation operations over these embeddings that assume compositionality.

2 RELATED WORK

Compositonality in linguistics was first defined back in 1892 by Frege [6] and later neatly placed into the present context by Janssen [8]. In 2010 the mathematical foundations of compositional semantics were described by Coecke et al. [4].

Using a simple mathematical operation on word representation vectors to compose them was done by Mitchell and Lapata [13]. This work does not yet use the real valued word embeddings which are popular today. However the work does compare various operations on word embeddings and how it affects their composition, similar to this work. They come to different conclusions however. In their results multiplicative models are superior to the additive models.

In regard to the deep learning approach for creation of word embeddings Mikolov et al. [11] talked about composition for word embeddings in terms of analogy tasks. They give a clear picture of the additive compositional properties of word embeddings, however the analogy tasks are still somewhat selective.

Combining syntactic structure, i.e. a parse, with real valued word representations in a recursive neural network parser was done by Socher et al. [16]. This work created a structure for composition using parse tree which were learned by a two-step model. The model relies on a correct parse to make a good compositions, this is not always the case. But it is the first model that tries to separate syntactic information from the word embedding to focus on the semantics. In other work Socher et al. [17] applied the parsing approach to composition for sentiment analysis.

The method by Faruqui and Dyer [5][1] is a popular way to evaluate word embeddings. Their evaluation combines 13 different word pair sets, with a total of 11,212 word pairs, and they use the Spearman's rank correlation coefficient as a metric. Because their method focuses on word pairs they can capture the semantic similarity between words, but can not necessarily say something about their compositionality.

3 DATASET

In order to test semantic composition we first turn to a dictionary for data. The words or lemmas in dictionaries all have good definitions, which are composed semantically into the meaning of that word, and thus ideal for our task. We choose to use WordNet[12] as the basis for our dataset. The synonym set in WordNet allows for the creation of pairs $x = (d, l_d)$ of definitions $d \in \mathcal{D}$ with one, or many lemmas $l_d \subset \mathcal{L}$ associated with that definition.

We only consider single word, i.e. unigram, lemmas for $\mathcal{L}$. A definition is a list of words where $d = \{w^d \in \mathcal{W} | w_1^d, w_2^d \dots w_n^d\}$. We make sure to remove stop words from this definition. If in the original WordNet synonym graph we find that the lemma is actually one of the definition words, we do not add that lemma to l_d.

Now that we have a vocabulary of definition words $\mathcal{W}$ as well as a vocabulary of target words $\mathcal{L}$, we find vector representations for each definition and target words from three popular large pre-trained word vector datasets: Word2Vec[2][11], GloVe[3][14] and fastText[2]. Each of these pre-trained word embedding sets has embeddings with a dimensionality of 300, as to keep our final comparison relatively fair.

Now we make sure to only keep words in our dataset that have representations for all three datasets. An intersection of the words from WordNet, Word2Vec, GloVe and fastText remains and this results in $|\mathcal{X}| = 66,464$ unique data points with a target vocabulary of $|\mathcal{L}| = 48,153$ unique lemmas.

This dataset in itself can be used to test composition in multiple ways, there are many different approaches to ours one could take. We think that Word2Vec, GloVe and fastText together is a good set of word embeddings, but one could easily add other embeddings for more fine grained experiments. Furthermore, one could use the dataset to tune existing embeddings to allow for better composition and test whether this improves the representations in general.

We made the code to create the entire dataset freely available[4].

4 METHODS OF COMPOSITION

In order to compose a meaningful representation from a set of words d that is close to the target representation of lemma l we first will apply simple mathematical operations. We chose to use +, $\times$, max(d) and average(d) over each of the 300 dimensions of the representation. In addition, we learn to compose a representation using an LSTM. This will allow the model to take order into account as well as better incorporate syntactic information, even though this approach has the same limitation an NMT encoder has.

4.1 Composing by operation

First we will try to create a composition by applying +, $\times$, max(d) and average(d). It should be no surprise that composing by simple mathematical operation is not ideal, since the act of composing does not consider the relationship between individual words as well as the order of words. Instead such relationships should already be present in the space of all words under the operation.

But by analyzing the results from simple operations we could have new insights into the word embedding space itself, how it already has compositional properties and how it can be used. For example the document representation model Doc2Vec uses a average(d) operation to compose embeddings of any kind. Our evaluation will show that for the three popular embeddings we choose this is not the optimal operation.

4.2 Learning to compose

In addition to simple mathematical operations we also trained a single layer bidirectional RNN model to compose embeddings. There are two big advantages to using an RNN over simple operations, because we now learn parameters for functions to improve the compositional representation we are not bound to the original embedding space and its compositional properties. Furthermore, a RNN will be able to take word order into account and thus can do more with syntactical information.

Our bidirectional RNN model uses the LSTM units for its hidden layer. Our vanilla model does not contain an attention mechanism and thus is just a plain bidirectional LSTM. It uses 300 hidden units and produces a representation which is directly optimized using Adam to minimize mean squared error between its produced

[1] More commonly known as wordvectors.org.

[2] Google made pre-trained vectors available on https://code.google.com/archive/p/word2vec/ from their Google News dataset of 100B tokens.

[3] For GloVe we used the representations from the Common Crawl which has 840B tokens and a vocabulary of 2.2M.

[4] You can create this dataset using the script from: *removed for anonymity* .

composed representation and the representation of the target word. We regularize using dropout with a probability of 0.1.

In future work one could improve this model or explore other methods for learning composition, such as a convolutional approach [7] or an approach which predicts weights for a weighted sum using the input representation as done in attention models.

5 EVALUATION

Instead of using a distance metric such as euclidean distance or cosine similarity we compare our composed representation to our target word representation using ranking with the Balltree algorithm for efficient nearest neighborer ranking. The algorithm produces a complete ranking of all the 48,153 target words. In the subsequent ranking we mark all target words from l_d as equally relevant, and all other words as not relevant. An important thing to note is that we do not normalize the vectors for one simple reason, this normalization does not occur in neural architectures and thus should not be done when we evaluate to make more informed decisions about architecture. This will result in different results for + and average(d), as opposed to them being the same with cosine similarity based ranking approaches.

We believe that ranking is superior to other evaluation methods because it is independent of the provided embedding space. Additionally there are several metrics you can compute out of a ranking which could be interpreted in different ways, especially regarding compositionality. Also ranking allows us to find structure in the very noisy compositional representations.

Now that we have obtained the ranking and the relevant results we can apply several well known ranking measures: Mean Reciprical Rank (MRR), Mean Average Precision (MAP) as well as Mean Precision@10 (MP@10). In addition we evaluate using Mean Normalized Rank (MNR) which describes the fraction of the total dataset that should be viewed before encountering a relevant result. For MNR and MRR we use the rank of the first relevant target word in l_d.

$$MNR := \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{\text{rank}_d}{|\mathcal{L}|} \qquad MRR := \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{\text{rank}_d} \tag{1}$$

Here is rank_d the rank of the first relevant target word for the definition d. Although MRR is more common, we found MNR to be better interpretable.

$$MAP := \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{\sum_{r_d=1}^{|\mathcal{L}|} P(r_d) \times \text{rel}(r_d)}{|l_d|} \tag{2}$$

MAP as well as MP@10 captures the possibility of multiple relevant target words into the metric, where MAP is a recall based metric and MP@10 is a precision based metric. For all metrics we can say: "higher is better", except for MNR where lower is better.

5.1 Quantitative results

Table 1 shows the results for our evaluation, and we can clearly see that additive composition, i.e. using the + operation is far superior to all other composition methods. Word2Vec performs relatively poorly on this compositional dataset, but this can be explained by the nature of the test data. The dataset where Word2Vec was trained on used news data, where fastText and GloVe use more definitional data, Wikipedia and Common Crawl respectively. fastText combined with additive composition will be able to compose definitions the best. We can see that GloVe performs better across compositional operations and this makes GloVe the more flexible alternative.

Table 1: The results from evaluating ranking. The results for MRR, MAP, and MP@10 are denoted in fractions of $\frac{1}{100}$.

		Word2Vec	fastText	GloVe
+	MNR	0.094	**0.075**	0.094
	MRR	4.983 %	6.852 %	2.734 %
	MAP	4.304 %	6.153 %	2.202 %
	MP@10	1.617 %	1.947 %	8.167 %
×	MNR	0.444	0.439	*0.342*
	MRR	0.107 %	0.196 %	0.154 %
	MAP	0.090 %	0.182 %	0.133 %
	MP@10	0.017 %	0.037 %	0.023 %
average(d)	MNR	0.287	0.210	*0.168*
	MRR	1.358 %	2.309 %	1.033 %
	MAP	1.143 %	2.020 %	0.798 %
	MP@10	0.270 %	0.530 %	0.243 %
max(d)	MNR	0.297	0.222	*0.180*
	MRR	0.841 %	2.065 %	0.796 %
	MAP	0.688 %	1.800 %	0.645 %
	MP@10	0.187 %	0.437 %	0.183 %
LSTM	MNR	0.376	0.309	*0.234*
	MRR	0.074 %	0.095 %	0.118 %
	MAP	0.069 %	0.084 %	0.098 %
	MP@10	0.017 %	0.006 %	0.020 %
Random	MNR	0.448		
	MRR	0.027 %		
	MAP	0.022 %		
	MP@10	0.003 %		

However the overall trend is that truly being good at composition is still hard for embeddings under simple operations. To highlight this we added a randomly generated ranking for all data points. In this random baseline we see a MAR of 0.448 and not of 0.500 because there are various data points with more than one relevant target words. When we look at the best MNR score for fastText with addition, 0.075 comes down to seeing 3,611 of the total 48,153 target words before encountering a relevant one, which is way better than the random baseline but still very bad. This is good news since it means there is a lot of room for improvement in making embeddings more composable. Another interesting thing to remark is the higher score for GloVe on MP@10 for the additive approach.

The results from our "Learning to Compose" approach are not as expected. Across datasets for metrics MRR, MAP and MP@10 it performs even worse than the × operation. Our hypothesis was that an LSTM model would be flexible enough to be able to learn a task at which simple addition is already relatively good at. However we see that this is not the case for complex word embeddings, and this

Table 2: These are some examples from the top-5 ranked results from the fastText embeddings. Other examples follow the same discussed patterns. This table also shows the excluded stop words, we italicized the actual definition tokens from d.

	Target Words l_d	Definition	$\|d\|$	RR
+	drydock	A *large dock* from which *water* can be *pumped* out; *used* for *building ships* or for *repairing* a *ship* below its *waterline*	10	1.0
×	kitten	Have *kittens*	1	1.0
	impolitic	Not *politic*	1	0.5
max(d)	fruitlessly, unproductively, unprofitably	In an *unproductive manner*	2	1.0
	imperialist, imperialistic	Of or *relating* to *imperialism*	2	1.0

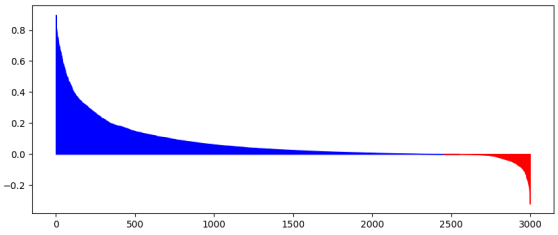

Figure 1: This figure shows the difference in MNR between + and average(d) on the GloVe data for 3000 definitions.

could be due to the multiplicative components of the LSTM module. When composing word embeddings within a neural network architecture this is another thing to consider.

5.2 Qualitative results

Table 2 shows some of the best ranking examples. The example from the table illustrate patterns that can be attributed to the mathematical operations applied. As we already saw in the quantitative analysis the multiplicative operation performs poorly. And we see this back in our example since the best ranking examples are not composed at all, i.e. their definition only exists of one word. The max(d) operation also shows an interesting pattern where we see definitions containing two words appear in the top ranked results, and one of these words is closely related to the target word. This is behavior you would expect of the max(d) operation. The additive operation shows some sophisticated composition examples to be its best ranking results, which is certainly interesting and requires further investigation.

To see at what types of sentences particular operations are actually good at we looked at the difference in metrics for specific definitions. In figure 1 we see the difference in MNR for + and average(d). When looking at the definitions which perform best with average(d) compared to + we see specific definitions with a high number of tokens of rare target words. An example: *"CF": "The most common congenital disease... "*. When looking at opposite, i.e. the best performing under + we see more general definitions but still for a high number of definition words. An example: *"Receptor": "An organ having nerve endings... "*.

6 CONCLUSION

In this work we introduced a new take on an existing dataset combined with combinatorial methods to evaluate word embeddings on their semantic compositionality. With this evaluation methods one can gain new insights in the applicability of word embeddings in neural networks and how one should approach defining a neural network architecture to best cope with semantic composition. The main finding of this work is that context-based pre-trained word embeddings are not compositional. In future work we will be investigating how we can tune embeddings to retain their discriminative properties and their lexical semantics but allow for better semantic composition.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. (2014).
[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. (2016).
[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. (2014).
[4] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. (2010).
[5] Manaal Faruqui and Chris Dyer. 2014. Community Evaluation and Exchange of Word Vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
[6] Gottlob Frege. 1892. On concept and object. (1892).
[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. (2017).
[8] Theo MV Janssen. 2001. Frege, contextuality and compositionality. *Journal of Logic, Language and Information* (2001).
[9] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.
[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. (2013).
[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.
[12] George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database. (1998).
[13] Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition.. In *ACL*.
[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.
[15] Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (Eds.). The Association for Computational Linguistics, 298–307. http://aclweb.org/anthology/D/D15/D15-1036.pdf
[16] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with Compositional Vector Grammars.. In *ACL*.

[17] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, and others. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*.
[18] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. (2014).