Bigrams and BiLSTMs
Two neural networks for sequential metaphor detection

Yuri Bizzoni
Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg.
firstname.lastname@gu.se

Mehdi Ghanimifard
Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg.
firstname.lastname@gu.se

Abstract

We present and compare two alternative deep neural architectures to perform word-level metaphor detection on text: a bi-LSTM model and a new structure based on recursive feed-forward concatenation of the input. We discuss different versions of such models and the effect that input manipulation - specifically, reducing the length of sentences and introducing concreteness scores for words - have on their performance. [1]

1 Paper's contribution

This paper describes our contribution to the shared task on metaphor detection published by NAACL 2018's First Workshop on Figurative Language Processing.

In this paper, we will:

1. Present and compare two neural network models, (1) a bidirectional recurrent neural networks for long distance compositions and (2) a novel bigram based model for local compositions.

2. Show the results of ablation experiments on these two models.

3. Present some input manipulations and feature enrichment to improve their performance.

The implementation code and additional supplementary material is available here: https://github.com/GU-CLASP/ocota

2 Introduction

Automatic metaphor detection is the task of automatically identifying metaphors in a text or dataset (Veale et al., 2016). Traditionally, the main approaches to this problem have been of two kinds: either a set of manually crafted rules was applied to a text, or a machine learning algorithm was trained on a source dataset to identify patterns of features identifying metaphoricity. In the latter case, typically used features were "psycholinguistics" features such as abstractness or imageability [2]; hypernym-hyponym coercions as modeled by resources like WordNet; sequence probabilities as given by language models; and semantic spaces or word embeddings. Similar trends can also be observed in works dealing with other figures of speech (Zhang and Gelernter, 2015).

The use of word embeddings in metaphor processing - both in detection and interpretation - is particularly widespread, and distributional semantic spaces may represent the single most consistently used "tool" in this task. Su et al. (2017) combine word embeddings and WordNet hypernym/hyponym information to detect nominal predicative metaphors of the kind "X is Y" and to select a more literal target - thus producing a paraphrase of the metaphor.

Shutova et al. (2017) use unsupervised and weakly supervised learning to detect metaphors, exploiting syntax-aware distributional word vectors.

Gong et al. (2017) use figurative language detection - sarcasm and metaphor - as a way to explore word vector compositionality and try to use simple cosine distance to tell metaphoric from literal sentences: a word being out of context in a sentence has a likelihood of being metaphoric.

The reason why semantic spaces are consis-

[1] The model product of this paper competed in The Workshop on Figurative Language's Shared Task with team name OCOTA.

[2] Recent trends tend to see metaphoricity as a nuanced rather than binary property, and to take into consideration the correlation between figurativity and affective scoring (Köper and im Walde, 2016), an umbrella term usually including four psycolinguistic properties: abstractness, arousal, imageability and valence (Köper and Im Walde, 2016).

Proceedings of the Workshop on Figurative Language Processing, pages 91–101
New Orleans, Louisiana, June 6, 2018.

tently used in metaphor detection lies in the conception that metaphor, like metonymy and other figures of speech (Nastase and Strube, 2009), is a mainly contextual phenomenon. In this view, a metaphor is fundamentally composed of two different semantic domains, in which one domain acts as source - and is used literally - while the other acts as target - and is used figuratively.

In this frame, semantic spaces appear to be a very flexible and powerful frame to model such semantic domains in terms of words' clustering and distributional similarity (Mohler et al., 2014). Also, semantic spaces are relatively easy to build and handle, giving them an advantage over more time-consuming resources, such as very large knowledge bases and "is A" bases from web corpora, as in Li et al. (2013).

Gutierrez et al. (2016) use the flexibility of word vectors to study the compositional nature of metaphors and the possibility of modeling it in a semantic space.

Tsvetkov et al. (2014) use distributional spaces, together with several other resources such as imageability scores and abstractness to detect metaphors in English and apply a transfer learning system through pivoting on bilingual dictionaries to detect metaphors in multiple language.

A composite approach using both distributional features and psycho-linguistics scores for lexical items is also used by Rai et al. (2016) to perform metaphor detection using conditional random fields.

Metaphor detection with semantic spaces has also been explored in a multimodal frame by Shutova et al. (2016), where systems using only text-based distributional vectors are compared against systems using distributional vectors enriched with visual information.

The link between distributional information and metaphors appears so relevant that some studies presenting new general distributional approaches have elected metaphor detection as a benchmark to test their models (Srivastava and Hovy, 2014), and studies using diversified sets of resources for their classifiers report that distributional vectors are the best performing single device to tackle metaphor detection (Köper and im Walde, 2016).

Finally, Bulat et al. (2017) present a different kind of semantic space, not context-based but attribute-based, to detect and generalize over metaphoric patterns. In such spaces, words are represented by the attributes of the concepts they represent, so that for example *ant* is represented by elements such as *an insect*, *is black* etc. The authors describe a system to map conventional distributional spaces to pre-existent attribute-based spaces and show that such approach helps detecting metaphoric bigrams.

A recent approach is that of using neural networks for metaphor detection with pretrained word embeddings initialization. Bizzoni et al. (2017) and Rei et al. (2017) proved that this is a valuable strategy to predict metaphoricity in datasets of bigrams without any extra contextual or explicit world knowledge representations. While Bizzoni et al. (2017) show how a simple fully connected neural network is able to learn pre-existing a dataset of metaphoric bigrams with high accuracy and to achieve a better performance than previous approaches, Rei et al. (2017) present an ad-hoc neural design able to compose and detect metaphoric bigrams in two different datasets.

Do Dinh and Gurevych (2016) apply a series of perceptrons to the Amsterdam Corpus combined with word embeddings and part-of-speech tagging, reaching a f-score of .56.

Interestingly, a similar approach - a combination of fully connected networks and pre-trained word embeddings - has also been used as a preprocessing step to metaphor detection, in order to learn word and sense abstractness scores to be used as features in a metaphor identification pipeline (Köper and im Walde, 2017).

3 Corpus

Metaphor processing suffers from a problem of data scarcity: annotated corpora for metaphor detection are relatively rare and of modest proportions.

In this work we will use the VU Amsterdam Metaphor Corpus (Krennmayr and Steen, 2017) train and test our models. To this date, the VU Amsterdam Metaphor Corpus (VUAMC) the largest publicly available annotated corpus for metaphor detection.

Metaphor corpora in other languages do exits, but, to the best of our knowledge, suffer of the same problem of data scarcity.

The VUAMC is divided into four sub-categories representing four different genres: news texts, fiction, academic texts and conversations. Every word in the corpus is manually annotated by sev-

eral annotators for metaphoricity. In the corpus, metaphor, simile and personification are equated, while also implicit metaphors are taken into consideration. For example, in the sentence *To embark on such a step is not necessarily to succeed immediately in realizing it* the word *it* is considered an implicit metaphor since it refers to the words *step* that was used metaphorically.

The corpus covers about 190,000 lexical units, randomly selected from the BNC Baby corpus.

According to Krennmayr and Steen (2017), the genre with a higher percentage of manually detected metaphors is academic texts (18.5%), followed by news (16.4%), fiction ("only" 11.9%) and conversation (7.7%). Given the very fine-grained nature of metaphor annotation applied to the corpus, the authors also find that the parts of speech that tend to be used metaphorically most often are prepositions and verbs, followed adjectives and nouns.

Due to its dimensions, diversity and accessibility, the VU Amsterdam Metaphor Corpus has been used in a number of studies. Using it can provide a direct comparison to important previous works and proposed models. This makes of the VUAMC a valuable resource for metaphor detection and processing.

Nonetheless, the VU Amsterdam Metaphor Corpus presents some difficulties: the semantic annotation of metaphor can be extremely fine-grained and cross the boundaries with word sense disambiguation.

For example, in the sentence:

> The 63-year-old head of Pembridge Investments, through which the bid is being mounted says, 'rule number one in this business is: the more luxurious the luncheon rooms at headquarters, the more inefficient the business'.[*a1e-fragment01-5*]

three words were annotated as metaphoric: *head*, *through*, *mounted*, *rule*, *in*, *this* and *headquarters*.

Sometimes the annotation itself can be puzzling or questionable. In the sentence:

> There are other things he has, on his own admission, not fully investigated, like the value of the DRG properties, or which part of the DRG business he would keep after the break up . [*a1e-fragment01-7*]

the following words are annotated as metaphoric: *things*, *on*, *admission*, *part*, *keep* and *after*.

While the very fine-grained metaphoricity of *things*, *part* and *keep* is to some extent still understandable - these terms are not used in their physical sense to indicate material objects, such as a concrete slice of something, or the act of physically keeping something with oneself - the metaphoric nature of *admission* remains quite opaque. At the same time, it is not clear why the annotators ignored the metaphoric interpretation of *the break up*.

There are also harder to explain examples, at least from our perspective. The sentence

> Going to bed with Jean fucking, fucking shite! [*kbd-fragment07-2586*]

is annotated as completely literal - no metaphoric usage is detected by the annotators.

In the sentence

> Take that fucking urbane look off your face and face reality, Adam [*fpb-fragment01-1343*]

the following words are annotated as metaphoric: *take*, *that*, *off*, *face*.

All the remaining terms have to be considered as literal, which looks slightly incoherent with the previous fine-grained metaphoricity annotations.

4 Models

4.1 Architectures

In this work we present two alternative neural architectures to process sentences as input and predict words' metaphoricity as output.

The first model we discuss is composed of a bi-directional LSTM (Schuster and Paliwal, 1997) and two fully connected or dense layers, having respectively dimensionality of 32, 20 and 1. We will also show results for deeper and more shallow alternative versions of this model.

Sun and Xie (2017) recently tried to tackle verb metaphor detection on the TroFi corpus (Birke and Sarkar, 2006) using Bi-LSTMs with word embeddings. For their study they tried different kinds of input: using the whole sentence; using a subsequence composed of the target verb and all its dependents; using a sub-sequence composed of the target verb, its subject and its object. Interestingly, they show that the simplest approach -

taking into consideration the whole sentence - returns the best results, with an F score only slightly lower than that achieved by a composite approach taking into consideration all of the previous different inputs together.

The main difference with our architecture is the presence of the final Perceptrons (fully connected networks). Sun and Xie (2017) don't mention further hidden layers beyond the bi-LSTM.

We also don't have any form of syntactic preprocessing and we only use the sequence of the standard word embeddings to represent the whole sentence. Finally, we are interested in considering the different performances of bi-LSTMs on different part-of-speech elements: metaphor recognition on functional words is supposedly harder, since these words have a more complex semantic signature in distributional spaces.

In this spirit we find worth it approaching the problem with a relatively "standard" neural framework.

The second model we discuss is a simple sequence of fully connected neural networks.

We present the design of this architecture in Figure 1.

This model is a generalization of neural architectures for bigram phrase compositions as tested on Adjective-Noun phrases in Bizzoni et al. (2017). While a similar approach is already attempted in Do Dinh and Gurevych (2016), we introduce a recursive variant which can make the compositions deeper and while allowing wide window sizes. There have been more sophisticated architectures such as Kalchbrenner et al. (2014), which take a similar approach for sentence representation with convolutional neural networks, but we propose a simpler method only using dense compositions.

We built our architecture using the Python library Keras (Chollet et al., 2015).

For both our models we used Adam optimizer.

4.2 Input manipulation

We compare two different features representations: 1. different word embeddings, 2. concreteness scores as word representations. In addition to ablation test for feature representations, we examined the effect of breaking sentences in shorter sequences.

Embeddings We tried two types of pre-trained word embeddings both with 300 dimensions: (1)

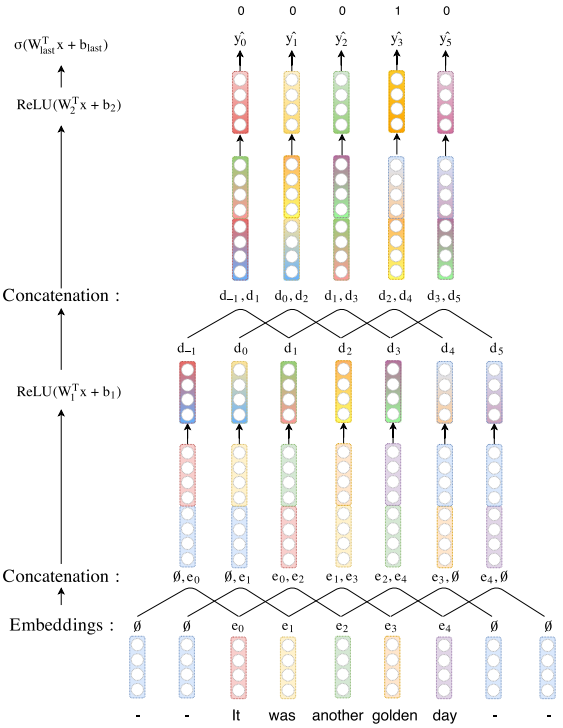

Figure 1: Bigram composition networks with depth $n = 2$.

GloVe (Pennington et al., 2014) (2) Word2Vec (Mikolov et al., 2013). Since these vector spaces are trained on different corpora, there are some out-of-vocabulary words, we represent these words with zero vectors. Additionally, Word2Vec is using a sub-sampling technique for more efficiency which consequently it doesn't cover most frequent words. In order to expand the word-coverage, we also trained GloVe embeddings on British National Corpus (Consortium et al., 2007) from which the VUAMC corpus was sampled, and compared it with both pre-trained Word2Vec embeddings on Google News corpus and standard GloVe embeddings trained on Common Crawl corpus.

Explicit features It has been observed in several works that metaphoricity judgments are partially related to a gap in concreteness between the target word and its context. Köper and im Walde (2017) try detecting all metaphoric verbs in the Amsterdam corpus using this single feature. Bizzoni et al. (2017) show how a network trained for metaphor detection on pairs of word embeddings can "side-learn" noun abstractness.

A metaphor functioning on this axis is composed of an abstract and a concrete element: in such case, usually, the concrete element is the

metaphoric one. The expression “In a window of 5 years, between 2011 and 2016” could be considered a metaphor playing on this level, where the more concrete word ”window” has a metaphoric sense.

There are kinds of metaphors functioning at different semantic levels: for example a synesthesia, which can be considered a sub-type of metaphor, is an expression where a word linked to a sensorial field is used to refer to a term that pertains to another sensorial field.

In this case, the features used metaphorically are usually on a similar level of abstractness. However, for our purposes the abstract-concrete features may be among the most important to take into consideration.

While the abstract-concrete polarity is represented in distributional embeddings, it is possible that taking such features more explicitly into consideration would help a neural classifier. Brysbaert et al. (2014) released a list of almost forty thousand English words annotated along the concrete-abstract axis, annotated by over four thousand participants.

We try using such scores as an extra dimension for the distributional embeddings: we thus obtain sequences of 301-dimensional embeddings, the last dimension being the human rating of concreteness. For the out-of-vocabulary words we use the average concreteness value of 2.5.

This resource allows us to assign to (almost) every word in the dataset an explicit concreteness score. When a word might have more than one sense, the annotations seem to use the most concrete one: for example the word “node” has a concreteness score of 4 out of 5. For comparison the words “output” and “literally” have a score of 2.48 and the word “being” has a score of 1.93.

It must be noted that the abstract-concrete gap is not necessarily the best way to describe the kind of metaphors represented in this specific corpus. The network should be able to mark as metaphoric words in this dataset that have a low level of concreteness, such as “approach” (2.76), in equally abstract contexts, such as “latest corporate reveals laid-back approach” (here “approach” was marked as metaphoric in VUAMC).

Many of the metaphoric uses outlined here are so ingrained in language that their actual concrete origins may be under-represented not only in modern day corpora, but even in many modern day annotators’ minds. We discussed various cases of this problem in the section about the corpus: words that have gradually assumed a new and main sense in the English language are often annotated as metaphors in the VUAMC.

Concreteness score window	number of words
1-2	38 262
2-3	36 730
3-4	28 664
4-5	14 473

Table 1: Concrete and abstract tokens in VUAM corpus according to Brysbaert et al. (2014) dataset.

Nonetheless, the abstract-concrete polarity remains one of the main semantic dimensions to interpret and understand metaphors and has been explicitly used in several metaphor detection tasks with promising results.

We can thus partly revert to feature engineering and see whether adding this dimension can improve the performance of our models.

Sentence breaking Including long sentences in our training dataset makes it necessary to consistently pad short sentences with zero-vectors. In our experiments we have seen that this seems to slow down and harm training for our models, since they will try to learn both patterns for sequences of pre-trained embeddings and patterns for long sequences of vectors filled with 0s.

To partly avoid this problem, we can break long sentences into two or more shorter elements. We assume that long distance information is not particularly important here to detect metaphoricity, while long padding can affect performance.

4.3 Preprocessing

We chose a maximum sentence length of 50: while the longest sentence in the dataset is 87 words, the vast majority of the elements in the dataset is less than 50 words long. Out of vocabulary words, which are words that did not have a corresponding vector in our embedding space, were replaced by a mock vector of all zeros. After shuffling the dataset, we use the first 1000 sentences of the corpus as test, and the rest of the data for training (11122 sentences). We used the same training and test data for all reported results.

4.4 Loss function

The design of the models is to predict the metaphoricity of each word in a sentence. The predicted value from a final layer with sigmoid activation is compared with the labeled data and usual logarithmic loss is used. However, most words do not have specified metaphoric or literal annotations in the dataset. Instead of assigning a non-metaphor value to unspecified tokens in a string, we modified the loss function in order to generate zero loss for these tokens.

4.5 Training

After shuffling the training data, 1000 samples are taken as holdout to find the overfitting point. With batch size 64 and and early stopping patience 3 based on validation loss we trained each model up to 15 epochs.

5 Results

5.1 Embeddings

Through a comparison of different semantic spaces, we found that the best performing space was GloVe trained on 42B Common Crawl, of dimensionality 300.

For the rest of our experiments we used these embeddings.

5.2 Baseline

In Table 2, we compare the results obtained from previous works on this task, and the performance of the "vanilla" settings of our model including a simple LSTM as our baselines. The comparison with Do Dinh and Gurevych (2016) shows that deploying deeper and more complex architectures on this set does not return particularly large improvements: we achieve an F1-score one point higher than Do Dinh and Gurevych (2016)'s results on a setting enriched with POS tags, and two points higher than the simplest model proposed in the paper.

It can be observed that our bigram composition architecture seems to produce comparable results considering the previous works. The influence of LSTM architectures appears thus further diminished.

Table 3 presents precision, recall and F-score values for several concatenation windows of our composition model. These results can be compared to the ones we obtain with deep Bi-LSTM models. Without external features such as concreteness or POS tagging, composing the input improves the model's performance up to a window of 3. Larger windows reduce the performance of the model.

In Table 4 we report the tests with different settings on depth and width of each layer.

It seems that widening the dimensionality of the Bi-LSTM itself beyond a certain limit does not improve - and rather harms - the model's performance in classification.

Regarding our first model, completely relying on the power of the Bi-LSTM architecture is not enough, and deeper fully connected layers are clearly playing a role.

We can also see that inserting a fully connected layer before the Bi-LSTM returns better results. This layer has a number of nodes as large as the number of dimensions of the input token embeddings. It can be another clue that the most relevant information for this task has to be searched in the word embeddings composing the sentence and their immediate surrounding, rather than in the structure of the whole sequence.

In conclusion, our results show that a quite standard deep neural architecture fed with good word embeddings can return promising results in metaphor detection. The "compositional" architecture also achieves comparable results, with an F score only a couple of points lower than that of the Bi-LSTM, indicating that "forcing" a network to give particular attention to the short or immediate context of each word in the data can improve its performance all the while reducing its depth, complexity and number of parameters. While this approach is not the one returning the absolute best F score, we consider the trade-off between its simplicity and its performance worth noting.

Our results also show a negative aspect: while we consider our models' performances encouraging, there is an ample room for improvement.

5.3 Feature experiments

Interestingly adding explicit semantic information such as concreteness ratings in our input - which means, somehow, reverting to feature engineering - did produce better results for the composition architecture, but not yet for our Bi-LSTM.

Table 5 show the results of our best performing models when the concreteness of the individual token was explicitly added to the embeddings.

Architecture	F1
Haagsma and Bjerva (2016)	.53
Do Dinh and Gurevych (2016)[3]	.56
Dense(1)	.22
LSTM(32)	.43
Bi-LSTM(32)	.46
Bi-LSTM(32)+Dense(20)	.50
Dense(300)+Bi-LSTM(32)+Dense(20)	**.56**
Concat(n=2)+Dense(300)	.55

Table 2: Performance of different models compared to the score reported by two relevant works in the literature. We report the performance of simpler models and their combinations as baselines. We used some abbreviations to describe the models in the table. For example, *Dense(1)* represents a single, fully connected layer of output length of 1, *LSTM(32)* is an LSTM with an output length of 32 and *Concat* represents our compositional model. Thus, *Concat(n=2)+Dense(300)* represents the bigram composition model with a concatenation window of 2 combined with a fully connected layer of 300 output units.

N	Precision	Recall	F1 score
1	.627	.459	.530
2	.588	.504	.543
3	.571	.531	**.550**
4	.649	.402	.497

Table 3: F1 for different windows of concatenation (N) in the composition model. N=1 is equivalent to no concatenation.

The results are higher than those returned by the same models trained and tested on the same sentences only with pre-trained distributional embeddings. It appears that simply adding the concreteness feature returns a better performance on the whole dataset. It is worth noting that in this case, and only in this case, the "compositional" architecture is the best performing, while the bi-LSTM has a harder time detecting metaphors in the textual data.

Finally, we try to break long sentences into shorter sequences, as we discussed in 4.2. The metaphors identified in the VUAM corpus do not generally require long-distance information to be detected. We can observe that this method improves the performance of our models: this is probably because the "noise" due to long padding of short sentences is reduced. Having less contextual information for words tagged as metaphoric or literal does not seem to have a real negative impact on the learning process.

As we show in Table 6, breaking sentences longer than 20 tokens into several short sequences reduces the number of misclassified elements in the set.

Not surprisingly, a combination of these two methods - adding explicit concreteness information and breaking long sentences - returns the best overall results, as can be seen in Table 7.

Finally, since these experiments were originally designed for the shared task in metaphor detection of the First Workshop in Figurative Language (NAACL 2018), in Table 8 we report our best performing models' results on the evaluation set provided in the task.

The last line reports the result from using both models together: as can be seen, the F score we get from taking into consideration the output of both architectures together is higher than the F score of the single models.

We can suppose that the two models are learning to detect slightly different kinds of metaphors - their true positives are not completely overlapping - and they can thus complement each other.

6 Conclusions

In the frame of NAACL 2018's shared task on metaphor detection, we explored two main approaches to detect metaphoricity through deep learning and compared their performances with different kinds of inputs. The overall single best performing system is a deep neural network composed of a bi-LSTM preceded and followed by fully connected layers, having access to concreteness scores for each token and running on relatively short sequences - thus reducing the effects of sentence padding.

We show that adding such features, our model is

Architecture	F1
Bi-LSTM(32)	.46
Bi-LSTM(32)+Dense(20)	.50
Bi-LSTM(400)+Dense(20)	.47
Bi-LSTM(32)+LSTM(32)+Dense(20)	.35
Bi-LSTM(400)+LSTM(32)+Dense(20)	.43
Dense(300)+Bi-LSTM(32)+Dense(20)	**.56**
Dense(300)+Bi-LSTM(300)+Dense(20)	**.56**
Dense(300)+Bi-LSTM(300)+LSTM(20)+Dense(20)	**.57**
Dense(300)+Bi-LSTM(300)+LSTM(100)+Dense(20)	.40

Table 4: Parameter tuning, testing both deeper and wider settings of the model. We write in parenthesis the dimensions each layer: for example *Dense(20)* is a fully connected layer with an output space of dimensionality 20.

N	Precision	Recall	F1
Dense(300)+Bi-LSTM(32)+Dense(20)	.642	.498	.561
Dense(301)+Bi-LSTM(32)+Dense(20)+Conc	.580	.491	.530
Concat(n=2)+Dense(300)+Conc	.554	.570	.562
Concat(n=3)+Dense(300)+Conc	.567	.593	**.580**

Table 5: Results for different models using embeddings enriched with explicit information regarding word concreteness. The first line works as baseline showing a model without input manipulation. *Concat(n=)* represents our compositional model, with *n*= representing the composition window length. *Conc* signifies the usage of concreteness scores. So for example *Concat(n=2)+Dense(300)+Conc* represents our compositional model with concatenation window of 2 combined with a fully connected layer of 300 output units and using the concreteness scores as additional information.

N	Precision	Recall	F1
Dense(300)+Bi-LSTM(32)+Dense(20)	.642	.498	.561
Dense(300)+Bi-LSTM(32)+Dense(20)+Chunk	.671	.570	**.621**
Concat(n=2)+Dense(300)+Chunk	.571	.561	.560
Concat(n=3)+Dense(300)+Chunk	.611	.400	.491

Table 6: Results for different models using sentence breaking to 20 (any sentence longer than 20 tokens is split in two parts treated as complete different sentences). The first line works as baseline showing a model without input manipulation. *Concat(n=)* represents our compositional model, *Chunk* signifies the usage of sentence breaking.

N	Precision	Recall	F1
Dense(300)+Bi-LSTM(32)+Dense(20)	.642	.498	.561
Dense(300)+Bi-LSTM(32)+Dense(20)+Chunk	.670	.571	.620
Dense(301)+Bi-LSTM(32)+Dense(20)+Conc	.581	.490	.531
Dense(301)+Bi-LSTM(32)+Dense(20)+Conc+Chunk	.649	.624	**.636**
Concat(n=3)+Dense(300)+Conc+Chunk	.632	.446	.523

Table 7: Results for different models using embeddings enriched with explicit information regarding word concreteness and sentence breaking to 20 (any sentence longer than 20 tokens is split in two parts treated as complete different sentences). The first lines work as baselines showing the performance of previous models (without any input manipulation, only chunking, only concreteness scores). *Concat(n=)* represents our compositional model, *Chunk* signifies the usage of sentence breaking, *Conc* represents the usage of concreteness scores.

N	Precision	Recall	F1
Dense(300)+Bi-LSTM(32)+Dense(20)	.638	.593	.615
Concat(n=2)+Dense(300)	.642	.498	.561
Combined results	.595	.680	**.635**

Table 8: Results for the evaluation set from the shared dataset competition (NAACL 2018). We used sentence breaking and concreteness information.

able to slightly outperform two baselines recently published.

We also found that combining these two systems gave the best results on the test set provided by the shared task.

Considering the difficult nature of the original annotations, we judge this a promising result. It could be the case that adding more explicit features further helps reduce the number of inconsistent detections on the corpus, but one of the goals of these experiments was that of keeping the feature engineering as contained as possible, reducing the number of external resources used to enrich the input.

We also explored a simpler neural architecture based on the recursive composition of word embeddings. Yielding a slighlty worse performance than the Bi-LSTM architecture, this model still shows that a much simpler architecture can reach interesting results.

7 Future Works

We think that an in depth error analysis of our models' shortcomings might represent an interesting contribution in order to better understand what neural networks are learning when they are learning metaphor detection. In future we would like to perform a systematic analysis of the errors of our networks both when used alone and when used in combination.

We would also like to extend the range of our comparisons to different, and simpler, machine learning algorithms to see to what extent the information provided in input - in terms of distributional information and explicit lexical scores - contributes to the performance of our models. While a consistent body of works on metaphor detection with "traditional" machine learning means already exists, we think that a direct comparison of our networks with other systems might help clarifying the contribution of deep learning to this task.

Acknowledgments

We are grateful to our colleagues in the Centre for Linguistic Theory and Studies in Probability (CLASP), FLoV, at the University of Gothenburg for useful discussion of some of the ideas presented in this paper.

We are also grateful to three anonymous reviewers for their several helpful comments on our earlier draft.

The research reported here was done at CLASP, which is supported by a 10 year research grant (grant 2014-39) from the Swedish Research Council.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Yuri Bizzoni, Stergios Chatzikyriakidis, and Mehdi Ghanimifard. 2017. " deep" learning: Detecting metaphoricity in adjective-noun pairs. In *Proceedings of the Workshop on Stylistic Variation*, pages 43–52.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 523–528.

François Chollet et al. 2015. Keras. https://github.com/keras-team/keras.

British National Corpus Consortium et al. 2007. British national corpus version 3 (bnc xml edition). *Distributed by Oxford University Computing Services on behalf of the BNC Consortium. Retrieved February*, 13:2012.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2017. Geometry of compositionality. In *AAAI*, pages 3202–3208.

E Dario Gutierrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin Bergen. 2016. Literal and metaphorical senses in compositional distributional semantic models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 183–193.

Hessel Haagsma and Johannes Bjerva. 2016. Detecting novel metaphor using selectional preference information. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 10–17.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 655–665.

Maximilian Köper and Sabine Schulte Im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350 000 german lemmas. In *LREC*.

Maximilian Köper and Sabine Schulte im Walde. 2016. Distinguishing literal and non-literal usage of german particle verbs. In *HLT-NAACL*, pages 353–362.

Maximilian Köper and Sabine Schulte im Walde. 2017. Improving verb metaphor detection by propagating abstractness to words, phrases and individual senses. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 24–30.

Tina Krennmayr and Gerard Steen. 2017. Vu amsterdam metaphor corpus. In *Handbook of Linguistic Annotation*, pages 1053–1071. Springer.

Hongsong Li, Kenny Q Zhu, and Haixun Wang. 2013. Data-driven metaphor recognition and explanation. *Transactions of the Association for Computational Linguistics*, 1:379–390.

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*, pages 3111–3119.

Michael Mohler, Bryan Rink, David B Bracewell, and Marc T Tomlinson. 2014. A novel distributional approach to multilingual conceptual metaphor recognition. In *COLING*, pages 1752–1763.

Vivi Nastase and Michael Strube. 2009. Combining collocations, lexical and encyclopedic knowledge for metonymy resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 910–918. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sunny Rai, Shampa Chakraverty, and Devendra K Tayal. 2016. Supervised metaphor detection using conditional random fields. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 18–27.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. *arXiv preprint arXiv:1709.00575*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 160–170.

Ekaterina Shutova, Lin Sun, Elkin Dario Gutierrez, Patricia Lichtenstein, and Srini Narayanan. 2017. Multilingual metaphor processing: Experiments with semi-supervised and unsupervised learning. *Computational Linguistics*.

Shashank Srivastava and Eduard Hovy. 2014. Vector space semantics with frequency-driven motifs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 634–643.

Chang Su, Shuman Huang, and Yijiang Chen. 2017. Automatic detection and interpretation of nominal metaphor based on the theory of meaning. *Neurocomputing*, 219:300–311.

Shichao Sun and Zhipeng Xie. 2017. Bilstm-based models for metaphor detection. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 431–442. Springer.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 248–258.

Tony Veale, Ekaterina Shutova, and Beata Beigman Klebanov. 2016. Metaphor: A computational perspective. *Synthesis Lectures on Human Language Technologies*, 9(1):1–160.

Wei Zhang and Judith Gelernter. 2015. Exploring metaphorical senses and word representations for identifying metonyms. *arXiv preprint arXiv:1508.04515*.