Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings

Corina Dima and Erhard Hinrichs
Collaborative Research Center 833 and Department of Linguistics
University of Tübingen, Germany
{corina.dima,erhard.hinrichs}@uni-tuebingen.de

Abstract

The present paper reports on the results of automatic noun compound interpretation for English using a deep neural network classifier and a selection of publicly available word embeddings to represent the individual compound constituents. The task at hand consists of identifying the semantic relation that holds between the constituents of a compound (e.g. WHOLE+PART_OR_MEMBER_OF in the case of '*robot arm*', LOCATION in the case of '*hillside home*'). The experiments reported in the present paper use the noun compound dataset described in Tratz (2011), a revised version of the dataset used by Tratz and Hovy (2010) for training their Maximum Entropy classifier. Our experiments yield results that are comparable to those reported in Tratz and Hovy (2010) in a cross-validation setting, but outperform their system on unseen compounds by a large margin.

1 Introduction

Recent research in computational semantics has increasingly made use of vector space representations of words in combination with deep neural network classifiers. This recent trend builds on the earlier successes of such representations and classifiers for morphological and syntactic NLP tasks (Collobert et al., 2011), and now also includes semantic tasks such as word similarity, word analogy as well as sentiment analysis (Mikolov et al., 2013; Pennington et al., 2014). The fact that the same type of vector representations can be initially trained for one or more NLP tasks and then be re-used and fine-tuned for a new, seemingly unrelated task suggests that such models can provide a unified architecture for NLP (Collobert and Weston, 2008). The fact that the performance of word embeddings, when combined with deep neural networks, improves in a multi-task learning scenario and can provide state of the art results for NLP further adds to the attractiveness of such methods.

One of the ways to further test the viability of such models and methods is to subject them to a wider range of well-studied NLP tasks and compare the results with previous studies on state-of-the-art NLP datasets.

One such task concerns the automatic interpretation of nominal compounds, a semantic phenomenon that has been widely studied in both theoretical and computational linguistics. This task consists of identifying the semantic relation that holds between the constituents of a compound (e.g. WHOLE+PART_OR_MEMBER_OF in the case of *robot arm*, LOCATION in the case of *hillside home*). Given that noun compounding is a highly productive word formation process in many natural languages, the semantic interpretation of compounds constitutes an important task for a variety of NLP tasks including machine translation, information retrieval, question answering, etc. Due to the productiveness of compounding, an adequate NLP system for the automatic interpretation of compounds will need to be able to generalize well to unseen data, i.e. to compounds that it has not been trained on. Vector space models that are based on very large training corpora and thus have a good coverage of the lexicon of a language provide a good foundation for achieving such generalization behavior. Novel compounds are typically formed of existing words in the language that are recombined to form a new complex word, whose meaning is usually more than the sum of the meaning of its constituents, but which are constrained by the combinatory potential

Proceedings of the 11th International Conference on Computational Semantics, pages 173–183,
London, UK, April 15-17 2015.

of a word. This combinatory potential, i.e. the tendencies to combine with other words, is exactly what is captured in a vector space model, since such models capture the sum of the contexts that a word typically appears in. Hence, vector space models and deep neural network classifiers appear to be well suited for experimenting with this task. Such experiments are facilitated by the availability of a large, annotated dataset of English compounds that is described in Tratz and Hovy (2010); Tratz (2011) and that was used in machine learning experiments.

The present paper reports on the results of experimenting with the Tratz (2011) dataset using four publicly available word embeddings for English and a deep neural network classifier implemented using the Torch7 scientific computing framework (Collobert et al., 2011). These experiments yield results that are comparable to those reported by Tratz and Hovy (2010) and by Tratz (2011), but outperform their system on unseen compounds by a large margin. The remainder of this paper is structured as follows: Section 2 presents previous work related to the automatic classification of compound relations. Sections 3 and 4 present the annotated noun compounds dataset and the four word embeddings that were used in the experiments. These experiments are summarized in Section 5. The paper concludes with a summary of the main results and an outlook towards future work.

2 Related Work

One of the earliest computational approaches to the classification of compound nouns is due to Lauer (1995), who reports an accuracy of 47% at predicting one of 8 possible prepositions using a set of 385 compounds. Rosario and Hearst (2001) obtain 60% accuracy at the task of predicting one of 18 relations using neural networks and a dataset of 1660 compounds. The domain-specific inventory they use was obtained through iterative refinement by considering a set of 2245 extracted compounds and looking for commonalities among them. Girju et al. (2005) use WordNet-based models and SVMs to classify nouns according to an inventory containing 35 semantic relations, and obtain accuracies ranging from 37% to 64%. Kim and Baldwin (2005) report 53% accuracy on the task of identifying one of 20 semantic relations using a WordNet-based similarity approach, given a dataset containing 2169 noun compounds. Ó Séaghdha and Copestake (2013) experiment with the dataset of 1443 compounds introduced in Ó Séaghdha (2008) and obtain 65.4% accuracy when predicting one of 6 possible classes using SVMs and a combination of various types of kernels. Tratz and Hovy (2010) classify English compounds using a new taxonomy with 43 semantic relations, and obtain 79.3% accuracy using a Maximum Entropy classifier and 79.4% accuracy using SVM^{multiclass} on their dataset comprising 17509 compounds and 63.6%(MaxEnt)/63.1%(SVM^{multiclass}) accuracy on the (Ó Séaghdha, 2008) data.

All these efforts have concentrated on English compounds, despite the fact that compounding is a pervasive linguistic phenomenon in many other languages. Recent work by Verhoeven et al. (2012) applied the guidelines proposed by Ó Séaghdha (2008) to annotate compounds in Dutch and Afrikaans with 6 category tags: BE, HAVE, IN, INST, ACTOR and ABOUT. The reported F-Scores are 47.8% on the 1447 compounds Dutch dataset and 51.1% on the 1439 compounds Afrikaans dataset.

3 The Tratz (2011) and Tratz and Hovy (2010) datasets

The experiments reported in this paper use the noun compound dataset described in Tratz (2011)[1]. This dataset, subsequently referred to as the Tratz dataset, is a revised version of the data used by Tratz and Hovy (2010) in their machine learning experiments, subsequently referred to as the Tratz and Hovy dataset. The Tratz dataset is the largest publicly-available annotated noun compound dataset, containing 19158 compounds annotated with 37 semantic relations. Table 1, which is an abbreviated version of Table 4.5 in Tratz (2011), illustrates these relations by characteristic examples and indicates the relative frequency of each relation within the dataset as a whole. The inventory of relations consists of seman-

[1]The dataset is available for download at `http://www.isi.edu/publications/licensed-sw/fanseparser/`

tic categories that resemble but are not identical to the inventories previously proposed by Barker and Szpakowicz (1998) and Girju et al. (2005). Tratz and Hovy (2010) motivate their new inventory by the necessity to achieve more reliable inter-annotator agreement than was obtained for these earlier inventories. The original Tratz and Hovy dataset consisted of 17509 compounds annotated with 43 semantic relations. Tratz (2011)'s motivation for creating a revised noun compound relation inventory with only 37 semantic relations was to create a better mapping between prepositional paraphrases and noun compound relations. The compound classification experiments described in Tratz and Hovy (2010) were, however, not re-run on the revised dataset. Since only the Tratz dataset is publicly available as part of the semantically-enriched parser described in Tratz (2011), this dataset was used in the experiments reported on in the present paper.

4 The embeddings

The automatic classification experiments presented in section 5 use a selection of publicly available word embeddings: the *CW* embeddings[2], decribed in Collobert et al. (2011), the *GloVe* embeddings[3], presented in Pennington et al. (2014), the *HPCA* embeddings[4], described in Lebret and Collobert (2014) and the *word2vec* embeddings[5] introduced by Mikolov et al. (2013). The vector size, the dictionary size, the amount of training data as well as the specific corpora used for creating each of these word embeddings are summarized in Table 2.

A word embedding $W : \mathcal{D} \rightarrow R^n$ is a function that assigns each word from the *embedding dictionary* $\mathcal{D}$ an n-dimensional, real-valued vector. The words in the dictionary $\mathcal{D}$ are *embedded* in a high-dimensional vector space, such that the representations of syntactically and/or semantically similar words are close together in the vector space. Word embeddings are the result of training language models on large amounts of unlabeled, textual data using various learning algorithms.

The *CW* embeddings (Collobert et al., 2011) were obtained by training a language model using a unified neural network architecture. The initial training step used unlabeled data (plain text from the support corpora) for training individual word embeddings. The training procedure uses a context window of size 11. Each context window is considered a positive training example for the word in the middle of the context window, which is called the *target* word. For each positive context window, a corresponding negative context window is generated where the target word is replaced with a random word from the dictionary. The training objective of the neural network can be described as learning to rank the correct context windows higher than the corrupted ones. This initial training step is followed by a supervised training step where the word embeddings are further refined in the context of 4 NLP tasks: part-of-sppech tagging, chunking, named entity recognition and semantic role labeling.

The GloVe model (Pennington et al., 2014) uses statistics of word occurences in a corpus as its primary source of information. It involves constructing a large co-occurence matrix X, where each entry X_ij corresponds to the number of times the word j occurs in the context of the word i. The sum of the elements on the i-th row of the matrix represents the number of co-occurences of the word i with any other word in the dictionary in a fixed-size context window (10 words to the left and 10 to the right of the target word). The model uses probability ratios as a mechanism for filtering out "irrelevant words" for a given word-word pair.

Lebret and Collobert (2014) generate the HPCA word embeddings by applying Hellinger PCA to the word co-occurence matrix, a simpler and faster method than training a full neural language model. Word frequencies are obtained by counting each time a word $w \in \mathcal{D}$ occurs after a context sequence of words T. The co-occurence matrix of size N x $|\mathcal{D}|$ contains the computed frequencies for all the words in the dictionary given all the N possible sequences of words. The 10,000 most frequent words in the

[2]The CW embeddings are part of the SENNA NLP suite which can be downloaded from `http://ronan.collobert.com/senna/`

[3]Available at `http://www-nlp.stanford.edu/projects/glove/`

[4]Available at `http://lebret.ch/words/`

[5]Available at `https://code.google.com/p/word2vec/`

Category name	Dataset percentage	Example
Objective		
OBJECTIVE	17.1%	leaf blower
Doer-Cause-Means		
SUBJECT	3.5%	police abuse
CREATOR-PROVIDER-CAUSE_OF	1.5%	ad revenue
JUSTIFICATION	0.3%	murder arrest
MEANS	1.5%	faith healer
Purpose/Activity Group		
PERFORM&ENGAGE_IN	11.5%	cooking pot
CREATE-PROVIDE-GENERATE-SELL	4.8%	nicotine patch
OBTAIN&ACCESS&SEEK	0.9%	shrimp boat
MITIGATE&OPPOSE	0.8%	flak jacket
ORGANIZE&SUPERVISE&AUTHORITY	1.6%	ethics authority
PURPOSE	1.9%	chicken spit
Ownership, Experience, Employment, Use		
OWNER-USER	2.1%	family estate
EXPERIENCER-OF-EXPERIENCE	0.5%	family greed
EMPLOYER	2.3%	team doctor
USER_RECIPIENT	1.0%	voter pamphlet
Temporal Group		
TIME-OF1	2.2%	night work
TIME-OF2	0.5%	birth date
Location and Whole+Part/Member of		
LOCATION	5.2%	hillside home
WHOLE+PART_OR_MEMBER_OF	1.7%	robot arm
Composition and Containment Group		
CONTAIN	1.2%	shoe box
SUBSTANCE-MATERIAL-INGREDIENT	2.6%	plastic bag
PART&MEMBER_OF_COLLECTION&CONFIG&SERIES	1.8%	truck convoy
VARIETY&GENUS_OF	0.1%	plant species
AMOUNT-OF	0.9%	traffic volume
Topic Group		
TOPIC	7.0%	travel story
TOPIC_OF_COGNITION&EMOTION	0.3%	auto fanatic
TOPIC_OF_EXPERT	0.7%	policy expert
Other Complements Group		
RELATIONAL-NOUN-COMPLEMENT	5.6%	eye shape
WHOLE+ATTRIBUTE&FEATURE &QUALITY_VALUE_IS_CHARACTERISTIC_OF	0.3%	earth tone
Attributive and Equative		
EQUATIVE	5.4%	fighter plane
ADJ-LIKE_NOUN	1.3%	core activity
PARTIAL_ATTRIBUTE_TRANSFER	0.3%	skeleton crew
MEASURE	4.2%	hour meeting
Other		
LEXICALIZED	0.8%	pig iron
OTHER	5.4%	contact lense
Personal*		
PERSONAL_NAME	0.5%	Ronald Reagan
PERSONAL_TITLE	0.5%	Gen. Eisenhower

Table 1: Semantic relation inventory used by the Tratz dataset - abbreviated version of Table 4.5 from Tratz (2011). Note that some relations have a slightly different name in the actual dataset than the aforementioned table; this table lists the relation names as found in the dataset.

Name	Embedding size	Dictionary size	Training data size	Support corpora
CW	50	130,000	0.85 bn	enWikipedia + Reuters RCV1
GloVe	300	400,000	42.00 bn	Common Crawl (42 bn)
HPCA	200	178,080	1.65 bn	enWikipedia + Reuters + WSJ
word2vec	300	3,000,000	100.00 bn	Google News dataset

Table 2: Overview of embedding sizes, dictionary sizes, training data sizes and support corpora for the four selected embeddings. The training data size is reported in billions of words.

dictionary were considered as context words, and size of the word sequence T was set to 1.

Mikolov et al. (2013) uses a continuous Skip-gram model to learn a distributed vector representation that captures both syntactic and semantic word relationships. The authors define the training objective of this model as the ability to find "word representations that are useful for predicting the surrounding words in a sentence or a document". The training context for a word is defined as c words to the left and to the right of the word.

For the *GloVe* and *HPCA* embeddings there are multiple sizes of word embeddings available: 50, 100, 200, 300 (6 billion words support corpus) and 300 dimensions (42 billion words support corpus) for *GloVe* and 50, 100 and 200 dimensions for *HPCA*. We have experimented with all the different sizes for each embedding and it was always the highest dimensional embedding that gave the best results in the cross-validation setup. Therefore, due to space limitations we only report results for the maximum size of each embedding.

5 The experiments

This section summarizes the experiments performed on the Tratz dataset using the four embeddings described in the previous section. Section 5.1 describes the pre-processing steps that had to be performed on the Tratz dataset in order to make it inter-operable with the embedding dictionaries. Section 5.2 describes the architecture of the classifier used in all the experiments. Section 5.3 presents the experiments performed using each of the embeddings individually as well as the best performing system that resulted from the concatenation of three out of the four selected word embeddings.

5.1 Dataset pre-processing

In order to make the best use of the word embeddings described in the previous section, several pre-processing steps had to be performed. The Tratz dataset contains about 1% training examples that are person names or titles starting with a capital letter, whereas such names appear in all lowercase in the embedding dictionaries[6]. Therefore all compound constituents of the Tratz dataset were converted to lowercase. This resulted in a *constituent dictionary* $\mathcal{C}$, $|\mathcal{C}| = 5242$ unique constituents for the entire Tratz dataset of 19158 compound instances.

The constituent dictionary $\mathcal{C}$ obtained from the Tratz dataset includes complex words such as '*health-care*' that are themselves compounds and which appear in the dataset as parts of larger compounds such as '*health-care legislation*'. Moreover, such complex words are not uniform in their spelling, appearing sometimes as a single word (e.g. '*healthcare*'), sometimes hyphenated (e.g. '*health-care*') and sometimes as two separate words (e.g. '*health care*'). Therefore such spelling variation had to be adapted to the spelling conventions used by each individual embedding. The same type of adaptation had to be performed in the case of misspelled words in the Tratz dataset and singular/plural forms of the same lemma. In cases where a constituent appears in the embedding dictionary as two separate words we use the average of the individual word embeddings as a representation for the constituent.

[6]On linguistic grounds, it is highly questionable whether personal names should be included in a compound dataset. However, since they are part of the Tratz dataset, we chose not remove them.

The Tratz dataset also contains some infrequent English words such as '*chintz*' (in '*chintz skirt*'), '*fastbreak*' (in '*fastbreak layup*') or '*secretin*' (in '*sham secretin*'), which are not part of the embeddings dictionaries. We used an unknown word embedding for representing such words. This embedding is already part of the dictionary for some embeddings (e.g. the *CW* embedding), and was obtained by averaging over the embeddings corresponding to the least frequent 1000 words for the other embeddings.

The pre-processed Tratz dataset was then partitioned into *train*, *dev* and *test* splits containing 13409, 1920 and 3829 noun compounds, respectively. The combined *train* and *dev* splits were also used to construct a 10-fold cross-validation set.

5.2 Classifier architecture

We used a deep neural network classifier implemented in the Torch7 scientific computing framework (Collobert et al., 2011) for the automatic classification of noun compounds. The classifier is trained in a supervised manner on the examples in the Tratz dataset. A training example pairs the two constituents of a compound (e.g. the individual words '*robot*' and '*arm*' in the case of '*robot arm*') with a semantic relation from the relation inventory (e.g. WHOLE+PART_OR_MEMBER_OF).

Figure 1 displays the architecture of the network which consists of four layers: an input layer, a lookup table, a hidden layer and an output layer. The lookup table (Figure 1a) is a $|\mathcal{C}|$xN matrix which contains an N-dimensional embedding for every entry in the constituent dictionary $\mathcal{C}$.

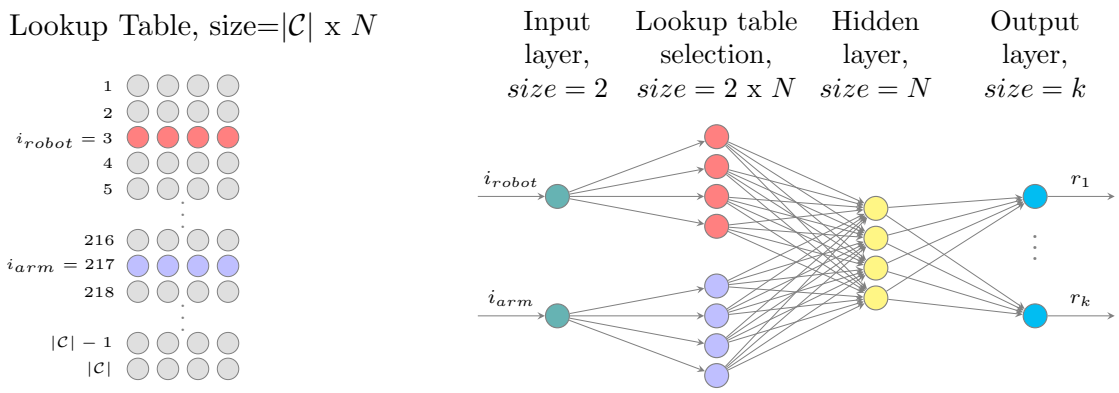

Figure 1: Classifier architecture

When a training example is presented to the network, the input layer and the lookup table are used to extract the word representations for the two constituents of the example compound. The input layer is used to input two numerical indices that uniquely identify each constituent in the constituent dictionary. These indices are then used to select the individual word representations of the compound constituents from the lookup table. The selected representations are concatenated and the combined representation is fed to the subsequent hidden layer. This hidden layer is intended to capture regularities in the data that are relevant for selecting the correct semantic relation of the example compound. Since the hidden layer is fully connected to the previous layer, such regularities can be drawn from the word representations of both compound constituents, as well as from the relative order of the two constituents. The optimal size of the hidden layer (N, which matches the size of the initial word representation) was determined empirically based on the *dev* split.

The resulting compound representation is then passed through a non-linearity (the logistic function, $\frac{1}{1+exp^{-x}}$) that maps it to the final output layer. The size of the output layer equals the number of semantic relations in the Tratz dataset. A softmax function is used to pick the semantic class which was assigned the highest score by the neural network classifier.

The purpose of the lookup table is to allow the generic word embeddings, which were constructed independently of the compound classification task, to be *fine-tuned* to the current task. This fine-tuning

is made possible by having the lookup table as an intermediate layer in the network – and thus modifiable by backpropagation during the training process – as opposed to having the embeddings directly as the input of the network. The fine-tuning of embeddings for a particular task, other than the one they have been initially trained for, has been advocated and proven effective by Collobert et al. (2011); Lebret and Collobert (2014) in the case of several NLP tasks like part-of-speech tagging, named entity recognition and sentiment analysis. In order to gauge the impact of embedding fine-tuning for the present task we compared the results of training a classifier with and without fine-tuning. The classifier without fine-tuning consists of an input layer of size 2 x N, a hidden layer of size N, and the output layer.

The network was trained using the negative log likelihood criterion, using averaged stochastic gradient descent optimization (Bottou, 2012) with a batch size of 5 and an initial learning rate of 0.9. The learning rate is adapted during the training phase, by setting it to 0.3 once the error on the development set is lower then a specified threshold. We used the *dev* split to choose the hyper-parameters of the model, which were used in all the reported experiments.

An early stopping criterion was employed in order to avoid the inherent tendency of neural networks to overfit the training data. We used a criterion proposed by Prechelt (1998), namely stop the training when the generalization error (i.e. the error on the *dev* set) has increased in s successive epochs. We set the number of successive epochs in which the generalization error is allowed to fluctuate to $s = 5$. The final model returned by the training procedure is the model with the best generalization error that was discovered during the training procedure.

5.3 Results

This section discusses the results of the experiments conducted with the neural network classifier presented in the previous section, using the four embeddings described in Section 4. The models are trained using the splits described in Section 3 in three setups: the DEV setup, where the model is trained on the *train* split and tested on the *dev* split; the CV setup, where the model is trained and tested using the 10-fold cross-validation set; the TEST setup, where the model is trained on the combined *train* and *dev* splits and tested on the *test* split. Each model is trained using two architectures: one using fine-tuning (DEV-F, CV-F, TEST-F) and one without fine-tuning (DEV-NO-F, CV-NO-F, TEST-NO-F). The results obtained for these three setups and these two architectures are summarized in Table 3. All the results in this table represent micro-averaged F1 measures[7].

Input embeddings	DEV-NO-F	DEV-F	CV-NO-F	CV-F	TEST-NO-F	TEST-F
CW-50	65.83	78.39	63.59	74.71	66.62	76.03
GloVe-300	74.17	77.81	72.89	76.57	76.05	75.87
HPCA-200	61.45	77.14	70.58	76.66	64.56	76.00
word2vec-300	71.46	73.54	69.07	71.93	71.38	71.59
random embeddings	-	74.17	-	64.54	-	71.43
CW-50+GloVe-300+HPCA-200	**79.01**	**79.48**	**76.20**	**77.70**	**78.14**	**77.12**

Table 3: Results for the task of automatic classification of noun compounds, using the same embeddings with two different architectures: (i) with fine-tuning (F) and (ii) without fine-tuning (NO-F)

The first four rows of Table 3 show the results obtained by the classifier when the individual compound constituents are represented using the four embeddings introduced in Section 4. The cross-validation setup provides the most representative results for the models trained on the Tratz dataset since it is based on different splits of data and averages the results of the individual folds. In all four cases, the models that perform embedding fine-tuning (CV-F) have consistently better results compared to the models that don't change the initial embeddings (CV-NO-F). The improvement brought about by fine-tuning is largest for the *CW* embedding (11.12 increase in F1 score). A plausible explanation for this

[7]The classification task at hand is an instance of *one-of* or *multinomial* classification, therefore micro-averaged F1 is the same as the accuracy.

improvement might be related to the size of the embedding. The *CW* embedding is much shorter than the other three embeddings, and hence the information in the embedding is therefore more coarse-grained. Fine-tuning such relatively coarse-grained word representations to the task at hand is then likely to yield a higher payoff. The importance of fine-tuning is further underscored by the fact that even with an initial random embedding the trained classifier is able to obtain an F1 score of 64.54 in the CV setup.

The classification results can be further improved by using word representations that are obtained by concatenating the word vectors from different embeddings. We experimented with using all the 11 possible combinations of the four embeddings (taken 2,3 and 4 at a time) as the initial representation of a constituent, and both architectures (with/without fine-tuning). The combination of all but the *word2vec* embedding as initial word representations, together with the fine-tuning architecture, empirically yielded the highest results (77.70 F1 score) in the cross-validation setup. Notice also that the size of the network grows considerably when the word representations of different embeddings are concatenated, not only for the input layers but also for the hidden layer, leading to a much more difficult training task when compared to more compact representations. This underscores the importance and effectiveness of good representations for the task at hand, since the performance of the classifier improves despite the more complex training task inherent in such larger networks. It also interesting to note that the training converges faster for the models that fine-tune the initial embeddings than for the ones that don't modify the input embeddings[8].

For completeness, we also report the results obtained for the DEV setup. These results are, as is to be expected, consistently higher than the ones obtained for the cross-validation setup. The improved performance on the DEV set can be explained by the fact that this setup was used to choose the hyper-parameters of the system (learning rate, batch size, optimization method, stopping criterion threshold). The improved performance could additionally be due to idiosyncrasies resulting from the particular split of the data.

The generalization capability to novel compounds of all the models can best be gauged by the results obtained on the unseen data provided by the TEST setup. The *test* split of the dataset contains 3829 compounds with a total of 2479 unique constituents. Almost a fifth of the the total number of constituents in the *test* split (18.23%) appear only in the *test* split, and were thus not seen by the classifier during the training process. As for the other DEV and CV setups, the fine-tuning process has the highest positive effect on the *CW* and *HPCA* embeddings and the performance of the combined embeddings once again outperforms the one of the individual embeddings. When comparing these results to those obtained for the CV setup, the most striking result is that there is almost no degradation in performance for the model with fine-tuning (77.12 TEST vs 77.70 CV F1 score) and actually an increase for the model without fine-tuning (78.14 TEST vs 76.20 CV F1 score).

The results of our experiments also support one empirical observation, namely that the impact of fine-tuning the embeddings decreases as the amount of data they have initially been trained on increases. The embeddings that gain the least from fine-tuning, or even perform slightly better without fine-tuning, are, in our experiments, the *GloVe* and the *word2vec* embeddings, as well as the combinations that include at least one of them. The common denominator of these embeddings is the large amount of data used for their initial training (between 40-100 times more training data than for the other embeddings, see Table 2). The embeddings trained on large support corpora seem therefore to be better suited for direct use in new tasks and have less to gain from fine-tuning, whereas the ones trained on small corpora perform considerably better if they are fine-tuned for new tasks.

Another key aspect underlined by our results is that a neural architecture in combination with the pre-trained, highly-informative word embeddings display a good generalization performance. Due to the high productivity of compounding, which leads to a constant stream of unseen data instances in real-world texts, such generalization ability is particularly important for the task at hand. This generalization ability is in part due to the nature of the word embeddings, which contain implicit information about the lexical semantics of all words in the embedding dictionary $\mathcal{D}$ trained by the initial language model.

[8]For example, in the Test setup, the models with fine-tuning stop, on average, after 15.2 training epochs, while the variants without fine-tuning stop only after an average of 38 training epochs.

While the constituent dictionary $\mathcal{C}$ used in the supervised training of the compound classification model is only a small subset of the original embedding dictionary $\mathcal{D}$ (e.g. 5242 vs 130000 words for the *CW* embedding), the lexical information about the remaining (e.g. 124 758) words is still implicitly present in the representation.

robot arm (H) WHOLE+PART_OR_MEMBER_OF	**plastic bag (H)** SUBSTANCE-MATERIAL-INGREDIENT	**birth date (H)** TIME-OF2	**hour meeting (H)** MEASURE	**cooking pot (H)** PERFORM&ENGAGE_IN	**hillside home (H)** LOCATION
dinosaur wing	leather bag	departure date	minute meeting	cooking fork	waterfront home
airplane wing	canvas bag	expiration date	week meeting	fishing pole	patio furniture
mouse skull	cardboard tray	release date	day meeting	recycling bin	fairway bunker
jet engine	gelatin capsule	expiry date	hour course	cooking liquid	beach house
airplane instrument	metal rack	election period	week course	drinking water	basement apartment
pant leg	wire rack	election date	month course	exercise room	ocean water
fighter wing	glass bottle	completion date	week conference	storage bin	cupboard door
bunny ear	tin plate	signing period	year course	fishing town	beach resort
goose wing	glass jar	launch date	hour journey	fishing village	bedroom door
shirt sleeve	wicker basket	redemption date	minute speech	feeding tube	basement vault

Table 4: Nearest neighbors of compounds from the Tratz dataset using **hidden layer representations**.

robot arm (I) WHOLE+PART_OR_MEMBER_OF	**plastic bag (I)** SUBSTANCE-MATERIAL-INGREDIENT	**birth date (I)** TIME-OF2	**hour meeting (I)** MEASURE	**cooking pot (I)** PERFORM&ENGAGE_IN	**hillside home (I)** LOCATION
robot spider	leather bag	delivery date	minute meeting	cooking spray	waterfront home
foot arm	plastic chain	payment date	day meeting	cooking fork	brick home
service arm	plastic pencil	birth weight	night meeting	cooking method	trailer home
mouse skull	garbage bag	election date	week meeting	flower pot	winter home
machine operator	canvas bag	publication date	weekend meeting	cooking liquid	boyhood home
elephant leg	plastic glove	release date	morning meeting	cooking class	retirement home
car body	diaper bag	departure date	afternoon meeting	cooking time	summer home
airplane wing	trash bag	redemption date	hour session	clay pot	family home
property arm	plastic sphere	completion date	evening meeting	cooking show	troop home
rocket system	glass bottle	retirement date	hour event	cooking demonstration	group home

Table 5: Nearest neighbors of compounds from the Tratz dataset using **initial embeddings**.

Additional evidence for the excellent generalization capability of the models reported on in this paper can be gleaned from inspecting the compound representations as constructed by the neural network at the hidden layer level. To this end we loaded the best performing model and removed the output layer, thus being able to isolate the representations constructed by the network for the Tratz compound dataset and to compute cosine similarity measures directly between the compound representations. Table 4 presents a selection of compounds from the Tratz dataset together with their closest 10 neighbors using the *hidden layer representations*. It is quite instructive to compare these 10 nearest neighbors with the 10 nearest neighbors from Table 5, obtained by computing cosine similarities with the *initial compound representations* (taken directly from the best performing combination of embeddings: CW-50+GloVe-300+HPCA-200). For the initial representations, similarity is largely restricted to compounds that share one of the constituents with the compound under consideration. The generalization potential of the initial embedding is thus largely restricted to partial string similarity (e.g. the nearest neighbors to '*cooking pot*' are compounds that contain either '*cooking*' or '*pot*' as one of their constituents). The neighbors obtained via the hidden layer representations display a much greater string variability, and at the same time a much stronger semantic similarity to the target compound. This semantic similarity manifests itself in a remarkable consistency in semantic relation (e.g. the PERFORM&ENGAGE_IN relation in '*cooking pot*' is shared by all of its 10 nearest neighbors when using the hidden layer representation). It is this combination of string variability and strong semantic similarity of the hidden layer representations that allows the network to generalize well to unseen data.

We conclude this section with a comparison of the results obtained for the present experiments with the results obtained by Tratz and Hovy (2010) using a Maximum Entropy (ME) classifier and a large number of boolean features. The features used to represent compounds in the Tratz and Hovy (2010) experiments are based on information from WordNet (synonym, hypernyms, gloss, etc.), from Roget's

Thesaurus, surface-level information (suffixes, prefixes) as well as term usage information in the form of n-grams. The ME classifier takes into account only the most useful 35000 features. Due to differences between the Tratz and Hovy dataset used by Tratz and Hovy (2010) and the Tratz dataset used in the present experiments, a direct comparison is not possible. These differences concern the number of data instances as well as the number of semantic relations – see Section 3 for a more detailed discussion. Such details notwithstanding, it is fair to say that our best result obtained in the cross-validation setting (77.70 F1 score) is close in performance to the reported state of the art (79.3% accuracy obtained by Tratz and Hovy (2010)) for this setting. However, our system outperforms that of Tratz and Hovy (2010) on the classification of unseen compounds by a wide margin (77.12 F1 score vs 51.0% accuracy). While the same disclaimer about the differences in the dataset used by Tratz and Hovy and in the present study applies for the unseen compounds in the *test* set, the fact that we obtained comparable results for the cross-validation (CV) and in the test (TEST) setups speaks well for the robustness of our model.

6 Conclusion

In this paper we have presented a deep neural network classifier approach for the task of automatic noun compound interpretation for English. We have shown that this approach achieves comparable results to the state of the art system trained on a closely-related dataset and significantly outperforms this earlier system when confronted with unseen compounds. Another advantage of our approach derives from the use of pre-trained word embeddings as word representations, rather than using large, manually selected feature sets that are constructed and optimized for a specific task as the initial word representations. Since word embeddings are more generic in nature and allow for re-training in a multi-task scenario, this approach has the potential of being reused, including for related tasks of semantic interpretation of compounds by prepositional paraphrasing, as has been proposed by Lauer (1995), or free paraphrasing, which has been the subject of a shared SemEval task (Hendrickx et al., 2013). However, for the time being, we have to leave such re-purposing to future research. Another direction for future research is to test our approach on other available noun compound datasets for English such as the one provided by Ó Séaghdha (2008). This would allow us to directly compare our approach to earlier systems trained on these datasets. Since the Tratz dataset is considerably larger than all the other publicly datasets, also testing our system on the latter and comparing the relative performance would allow us to better estimate the impact of the training data size as well as the one of the annotation scheme when training deep neural network classifiers for automatic noun compound interpretation.

Acknowledgements

The authors would like to thank the anonymous reviewers for their very useful suggestions. Financial support for the research reported in this paper was provided by the German Research Foundation (DFG) as part of the Collaborative Research Center "Emergence of Meaning" (SFB 833) and by the German Ministry of Education and Technology (BMBF) as part of the research grant CLARIN-D.

References

Barker, K. and S. Szpakowicz (1998). Semi-automatic recognition of noun modifier relationships. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pp. 421–436. Springer.

Collobert, R., K. Kavukcuoglu, and C. Farabet (2011). Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, Number EPFL-CONF-192376.

Collobert, R. and J. Weston (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research 12*, 2493–2537.

Girju, R., D. Moldovan, M. Tatu, and D. Antohe (2005). On the semantics of noun compounds. *Computer Speech and Language 19*(4), 479–496.

Hendrickx, I., P. Nakov, S. Szpakowicz, Z. Kozareva, D. O. Séaghdha, and T. Veale (2013). SemEval-2013 Task 4: Free paraphrases of noun compounds. *Atlanta, Georgia, USA*, 138.

Kim, S. N. and T. Baldwin (2005). Automatic interpretation of noun compounds using WordNet similarity. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*.

Lauer, M. (1995). *Designing Statistical Language Learners: Experiments on Compound Nouns*. Ph. D. thesis, Macquarie University.

Lebret, R. and R. Collobert (2014). Word embeddings through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, Gothenburg, Sweden, pp. 482–490. Association for Computational Linguistics.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Ó Séaghdha, D. (2008). *Learning compound noun semantics*. Ph. D. thesis, Computer Laboratory, University of Cambridge. Published as University of Cambridge Computer Laboratory Technical Report 735.

Ó Séaghdha, D. and A. Copestake (2013). Interpreting compound nouns with kernel methods. *Natural Language Engineering 19*(03), 331–356.

Pennington, J., R. Socher, and C. D. Manning (2014). GloVe: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, Volume 12.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pp. 55–69. Springer.

Rosario, B. and M. Hearst (2001). Classifying the semantic relations in noun compounds. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Tratz, S. (2011). *Semantically-enriched parsing for natural language understanding*. Ph. D. thesis, University of Southern California.

Tratz, S. and E. Hovy (2010). A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Verhoeven, B., W. Daelemans, and G. B. van Huyssteen (2012). Classification of Noun-Noun Compound Semantics in Dutch and Afrikaans. In *Proceedings of the Twenty-Third Annual Symposium of the Pattern Recognition Association of South Africa*, Pretoria, South Africa, pp. 121–125.