Reverse-engineering Language: A Study on the Semantic Compositionality of German Compounds

Corina Dima
Collaborative Research Center 833
University of Tübingen, Germany
corina.dima@uni-tuebingen.de

Abstract

In this paper we analyze the performance of different composition models on a large dataset of German compound nouns. Given a vector space model for the German language, we try to reconstruct the observed representation (the corpus-estimated vector) of a compound by composing the observed representations of its two immediate constituents. We explore the composition models proposed in the literature and also present a new, simple model that achieves the best performance on our dataset.

1 Introduction

Vector space models of language like the ones presented in (Collobert et al., 2011b; Mikolov et al., 2013; Pennington et al., 2014) create good representations for the *individual words* of a language. However, the words in a language can be combined into infinitely many distinct, well-formed *phrases* and *sentences*. Creating meaningful, reusable representations for such longer word sequences is still an open problem.

In this paper we focus on building representations for syntactic units *just above the word level*, by exploring compositional models for *compounds*. Bauer (2001) defines a compound as "a lexical unit made up of two or more elements, each of which can function as a lexeme independent of the other(s) in other contexts" (e.g. *apple tree*). The vast majority of compounds are compositional, i.e. we can understand the meaning of the compound if we know the meaning of its constituent words. We would like to equip the vector space model with a composition function able to construct a *composite* representation for *apple tree* from the representations of *apple* and *tree*. The composite representation should ideally be indistinguishable from its *observed* representation, i.e. the representation learned directly by the language model if the compound is part of the dictionary.

We situate our investigations in the context of the German language, a language where compounds represent an important fraction of the vocabulary. Baroni et al. (2002) analyzed the 28 million words German APA news corpus and discovered that compounds account for 47% of the word types but only 7% of the overall token count, with 83% of compounds having a corpus frequency of 5 or lower. The high productivity of the compounding process makes the compositional approach the most tractable way to create meaningful representations for all the compounds that have been or will be coined by the speakers of the German language.

German compounds have a strategic advantage for our study: they are generally written as a contiguous word, irrespective of how many constituents they have. Our example English compound, *apple tree*, translates into the German compound *Apfelbaum*, with the head *Baum* "tree" and the modifier *Apfel* "apple". Because the compound is written as a single word, we can directly learn the representations for the compound and for its constituents. Given a large dataset of German compounds together with their immediate constituents, and the corresponding distributed representations for each of the individual words, one can try to reverse-engineer the compounding process and learn the parameters of a function that combines the representation of the constituents into the representation of the compound. More formally, we are interested in learning a composition function f such that

$$c^{comp} = f(m^{obs}, h^{obs})$$

where $c^{comp} \in \mathbb{R}^n$ is the composite representation of the compound and $m^{obs}, h^{obs} \in \mathbb{R}^n$ are the observed representations of its modifier and its head. The function should minimize J, the mean squared error between the composite (c^{comp}) and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1637–1642,
Lisbon, Portugal, 17-21 September 2015. ©2015 Association for Computational Linguistics.

the observed (c^{obs}) representations of the $|C|$ compounds in the training set:

$$J = \sum_{i=1}^{|C|} \frac{1}{n} \sum_{j=1}^{n} (c_{ij}^{comp} - c_{ij}^{obs})^2$$

Several compositionality models have already been proposed in the literature (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012). In this paper we evaluate several of the proposed composition functions and also present a new composition model which outperforms all previous models on a dataset of German compounds.

2 Word Representations and Compounds Dataset

We trained 4 vector space language models for German (with 50, 100, 200 and 300 dimensions respectively) using the GloVe package (Pennington et al., 2014) and a 10 billion token raw-text corpus extracted from the DECOW14AX corpus (Schäfer, 2015). We use a vocabulary of 1,029,270 (1M) words, obtained by selecting all the words with a minimum frequency of 100 (the full vocabulary had 50M unique words). We used the default GloVe training parameters, the only modifications being the use of a symmetric context when constructing the co-occurence matrix (10 words to the left and to the right of the target word) and training each model for 15 iterations. All the vector spaces were normalized to the $L2$-norm, first across features then across samples using *scikit-learn* (Pedregosa et al., 2011).

The German compounds dataset used in the experiments is a subset of the 54759 compounds available in GermaNet 9.0[1]. The compounds in the list were automatically split and manually post-corrected (Henrich and Hinrichs, 2011). Each entry in the list is a triple of the form (compound, modifier, head). We filtered the entries in the list, keeping only those where all three words have a minimum frequency of 500 in the support corpus used to create the vector space representations. The reason for the filtering step is that a "well-learned" representation (based on a sufficiently large number of contexts) should allow for a more accurate reconstruction than a representation based only on a few contexts. The filtered dataset contains 34497 entries. This dataset was randomized and partitioned into `train`, `test` and `dev` splits according to the 70-20-10 rule. The dataset contains 8580 unique modifiers and heads, and a dictionary of 41732 unique words. 1345 compounds appear as the modifier or head of another compound.

3 12 ways to Represent A Compound

We adopt a notation similar to the one introduced in (Mitchell and Lapata, 2010), where the composite representation p is the result of applying a composition function f to the vectors u and v. In this study we tested the following composition functions:

1. $p = v$, the second constituent of the compound
2. $p = u$, the first constituent of the compound
3. $p = u \odot v$, component-wise vector multiplication
4. $p = (u \cdot u)v + (\lambda - 1)(u \cdot v)u$, dilation
5. $p = 0.5u + 0.5v$, vector addition
6. $p = \lambda u + \beta v$, weighted vector addition, where the λ and β are estimated using the training set. Models 1 through 6 were introduced in (Mitchell and Lapata, 2010).
7. $p = Uv$, where $v \in \mathbb{R}^n$ is the vectorial representation of the head word (given) and $U \in \mathbb{R}^{n \times n}$ is a matrix representation for the modifier, estimated with the help of the training data. The model estimates one matrix for each word that is used as a modifier. Referred to as *alm* in (Baroni and Zamparelli, 2010) and as *Lexfunc* in (Dinu et al., 2013b).
8. $p = M_1 u + M_2 v$, where $M_1, M_2 \in \mathbb{R}^{n \times n}$ are two matrices that modify the first and the second constituent vectors, respectively. In contrast to the previous model, this model estimates just one matrix for all the modifiers and one matrix for all the head words. Refered to as EAM in (Zanzotto et al., 2010) and as *Fulladd* in (Dinu et al., 2013b).
9. $p = g(W[u; v])$, where: $[u; v] \in \mathbb{R}^{2n \times 1}$ is the concatenation of the individual word vectors; $W \in \mathbb{R}^{n \times 2n}$ is a global matrix that: (i) combines the individual dimensions of the concatenated input vector $[u; v]$; (ii) brings the composite representation back into the $\mathbb{R}^{n \times 1}$ space; g is an element-wise function, in our experiments the hyperbolic tangent *tanh*. Introduced in (Socher et al., 2010).
10. $p = g(W[Vu; Uv])$. Introduced in (Socher

[1] http://www.sfs.uni-tuebingen.de/lsd/compounds.shtml

et al., 2012), it is a generalization of model 7. Each word is represented using an $\mathbb{R}^{n \times n}$ matrix and a $\mathbb{R}^n$ vector. The vectors are given, while the matrices are estimated using the training data. Referred to as *Fulllex* in (Dinu et al., 2013b).

11. $p = u \odot u' + v \odot v''$, the additive mask model (*Addmask*) and
12. $p = g(W[u \odot u'; v \odot v''])$, the global matrix mask model (*Wmask*), both presented in subsection 3.1.

Models 1 through 8 were tested using the implementations available in the DISSECT toolkit (Dinu et al., 2013a). As a side note, the *Lexfunc* implementation in DISSECT does not produce a composite representation for 11.5% of the our test data, where a word does not appear as a modifier during training. Therefore, we reimplemented the *Lexfunc* model and solved the missing training material problem by initializing the matrix for all the words in the dictionary with $I + \epsilon$, the identity matrix plus a small amount of Gaussian noise. This type of initialization was proposed by (Socher et al., 2012), and allows the model to back-off to the model $p = v$ when there is no data to estimate the parameters of the modifier matrix. We also reimplemented models 9 and 10, which were used in (Socher et al., 2010; Socher et al., 2012), as the existing implementations are part of a more complex recursive architecture aimed at constructing representations for full sentences.

3.1 The *mask* models

The newly introduced mask models build upon the idea that when a word w enters a composition process, there is some variation in its meaning depending on whether it is the first or the second element of the composition. Think, for instance, of the compounds *company* ***car*** and ***car*** *factory*. In the first case, *car* has its primary denotation, that of a road vehicle. In the second case, what matters more about the *car* is its *product* aspect, the fact that it is an “artifact produced in a factory”. A good representation of the word *car* should encode both aspects. Likewise, a good composition model should be able to select from the individual word representations only those aspects that are relevant for the composition process.

We want to give the composition model the possibility to deal with these slight sense variations, so we train, for each word in the dictionary, two *masks*, one for the case when it is the first word in the composition process and one for when it is the second word. The *masks* of the word w represented by $u \in \mathbb{R}^n$ are two vectors $u', u'' \in \mathbb{R}^n$. The mask vectors are initialized with a vector of all ones, $\mathbf{1}$, and estimated with the help of the training data. Each time w is the first word in the composition process, it is represented as the element-wise multiplication of the vector u and the mask u', $u \odot u'$. When w is the second word in the composition, it is represented by the element-wise multiplication of u and the mask u'', $u \odot u''$.

It is important to note that the initial vector representations remain fixed during the learning process. The learning process only affects the mask vectors. The composite representation of a compound like *car factory* is obtained by combining the masked representations, $u_{car} \odot u'_{car}$ and $v_{factory} \odot v''_{factory}$. We tried two different combination methods: (i) $p = u \odot u' + v \odot v''$, called *Addmask* (model 11), where the masked representations are combined via component-wise addition, and (ii) $p = g(W[u \odot u'; v \odot v''])$, called *Wmask* (model 12), where the combination of the masked representations is made via a global matrix $W \in \mathbb{R}^{n \times 2n}$ and a nonlinearity g (*tanh*), similar to model 10.

3.2 Implementing composition models

Models 7, 9 and 10 and the mask models were implemented using neural network architectures in the Torch7 library (Collobert et al., 2011a). We use the mean squared error as a training criterion, and optimize all models using *Adagrad* (Duchi et al., 2011) and a mini-batch of 100 samples. The hyperparameters were chosen by testing different parameter values and evaluating their performance on the `dev` set. To avoid overfitting we used early stopping (Prechelt, 1998). All the implemented models keep the input vectors fixed during the composition process.

Training the mask models entails estimating modifier and head masks for every word in the dictionary $\mathcal{D}$. The two types of masks to be learned can be formalized as two matrices $W_M, W_H \in \mathbb{R}^{n \times |\mathcal{D}|}$, where n is the size of the initial word representations. The masks of the word $w_i \in \mathcal{D}$ are the i^{th} rows in W_M and W_H. In Torch7 such representations can be learned using *lookup table layers* (Collobert et al., 2011b), which map matrix indices to the corresponding row vector.

The masked representation of the modifier is obtained by first feeding the index of the word to LT_{W_M}, the modifier lookup table, to obtain the modifier mask, and then multiplying the modifier mask with the initial representation for the modifier. The masked representation of the head is obtained in a similar manner via a lookup operation in LT_{W_H}, the head lookup table. The *Addmask* and *Wmask* models differ only in the composition method used after the masking process: the masked representations are directly added together in the case of *Addmask* and are passed through a composition matrix $W \in \mathbb{R}^{n \times 2n}$ and a nonlinearity g in the case of *Wmask*. The two matrices W_M, W_H are initialized with all ones and are modified via backpropagation during the training process.

4 Evaluation and Results

The twelve composition models presented in Section 3 were evaluated using word representations of increasing size (described in Section 2). All the models are trained on the `train` split and tested on the `test` split. We used the rank evaluation method proposed by (Baroni and Zamparelli, 2010) for a similar task: first, we generate a composite representation for each of the 6901 compounds in the test set; then, we use the cosine similarity to rank each composite representation with respect to the observed representations of the 41732 unique words in the dataset dictionary. If the observed representation is the nearest neighbour, the composition is assigned the rank 1. Similar to (Baroni and Zamparelli, 2010), we assign the rank 1000 ($\geq$1K) when the observed representation is not one of the nearest 1000 neighbours of the composite representation. We then compute the first, second and third quartiles (Q1, Q2, Q3) across all the compounds in the test set. A Q1 value of 2 means that the first 25% of the data was only assigned ranks 1 and 2. Similarly, Q2 and Q3 refer to the ranks assigned to the first 50% and 75% of data, respectively. The results of our evaluation are displayed in Table 1.

The observed representation of the head (model 1) was used as a strong baseline for the compound composition task. Two of the tested models, multiplicative (model 3) and dilation (model 4) score worse than the head baseline, while the additive models (5 and 6) score only slightly above it. The fact that the worst performing model is the multiplicative model is surprising considering its good performance in previous studies (Mitchell and Lapata, 2010). This might be either a side-effect of the normalization procedure, or a genuine incompatibility of this compositionality model with the vectorial representations produced by GloVe.

The new *Addmask* and *Wmask* models (introduced in Section 3.1) perform very well, with *Wmask* producing the best results on the test dataset across all dimensions. It is interesting to note that the linguistically motivated *Lexfunc* and *Fulllex* models, which build dedicated representations for each individual constituent, are outperformed by a simple model like *Fulladd*, that only learns two modification matrices, one for each position. The explanation is, in our opinion, that the available training material is not enough for training all the parameters of the complex *Lexfunc* and *Fulllex* models, but good enough for the more simple *Fulladd*.

The mask models are computationally cheaper than models like *Lexfunc* and *Fulllex*, as they they only train $2n$ parameters for each word in the vocabulary, and not n^2 parameters like the aforementioned models. They manage to strike a balance and learn a dedicated representation for each constituent with a small number of parameters, thus performing better than the more complex models.

We used non-parametric statistical tests to detect significant differences between the results obtained by the models. We focused our analysis on the best performing 4 models: model 9, which we will label the *Matrix* model, *Fulladd* (model 8), *Addmask* (model 11) and *Wmask* (model 12). The comparison takes into account two separate factors: (i) differences between the models using representations of the same size; (ii) differences in the performance of the same model using representations of different sizes.

A Friedman test on the ranks obtained by the 4 selected models on representations of size 300 showed that there is a significant difference between the models ($p < 0.01$). Pairwise comparisons (using the Wilcoxon signed rank test and Bonferroni corrections) showed that there is a significant difference ($p < 0.01$) between all but one pair of models, namely the *Matrix* and the *Addmask* models ($p = 0.9$). The same test confirmed that there are significant differences in the performance of the best model *Wmask* when using representations of different sizes ($p < 0.01$). Pairwise

no	**f**	**I**	**50d**			**100d**			**200d**			**300d**		
			Q1	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**
1	$p = v$	D	66	445	≥1K	36	202	≥1K	33	197	989	29	174	884
2	$p = u$	D	445	≥1K	≥1K	215	≥1K	≥1K	171	917	≥1K	144	808	≥1K
3	$p = u \odot v$	D	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K
4	$p = (u \cdot u)v + (\lambda - 1)(u \cdot v)u$	D	75	492	≥1K	38	213	≥1K	35	209	≥1K	30	181	926
5	$p = 0.5u + 0.5v$	D	85	408	≥1K	29	137	600.5	28	140	637	24	120	553
6	$p = \lambda u + \beta v$	D	62	329	≥1K	23.5	118	556	23	121	568	20	105	503
7	$p = Uv$ (on 88.5% of test data)	D	38	415	≥1K	15	147	≥1K	9	61	636	8	47	443
7	$p = Uv$	R	10	88	829	8	64	595	7	48	479.5	7	51	526.5
10	$p = g(W[Vu; Uv])$	R	3	18	178	3	12	111	3	16	188	4	26	334
9	$p = g(W[u; v])$	R	4	19	137	3	11	64	**2**	7	33	**2**	7	29
11	$p = u \odot u' + v \odot v''$	N	3	12	85	3	8	45	3	7	30	3	7	27
8	$p = M_1 u + M_2 v$	D	4	19	135	3	10	61	**2**	7	33	**2**	**6**	27
12	$p = g(W[u \odot u'; v \odot v''])$	N	**2**	**9**	**62**	**2**	**7**	**35**	**2**	**6**	**25**	**2**	**6**	**24**

Table 1: Quartiles for the 6901 composite representations in the test set, ranked with respect to the observed representations. Best possible rank is 1. D marks the models tested with DISSECT, R marks reimplementations of existing models and N marks new models.

comparisons showed that *Wmask* model significantly improves its performance ($p < 0.01$) when using word representations of increasing size (50, 100, 200 and 300 dimensions).

The twelve composition models were also compared in terms of the mean squared error (MSE) objective function, by computing the MSE between the composite and the observed representation of the compounds in the test set. The best scoring models in the rank evaluation were also the best in the MSE evaluation. However, the difference in performance between the best and the worst models was considerably smaller: the MSE of the multiplicative model is only twice as large as the MSE of the best performing *Wmask* model. This is in contrast to the rank evaluation where the multiplicative model assigned the observed representations in the test set only ranks ≥ 1000, while *Wmask* assigned ranks $\leq$ 25 to 75% of the test data. Additional investigations are necessary to estimate the impact of different objective functions on the performance of compositional models.

5 Comparison to related work

The experiments reported in this paper are, to the best of our knowledge, the first large scale experiments on the composition of German compounds. Other studies (Kisselew et al., 2015; Lazaridou et al., 2013) focused on morphologically complex words in German and English respectively. In terms of the size of the training and test material, our experiments are closest to the adjective-noun experiments in (Baroni and Zamparelli, 2010) and (Dinu et al., 2013b) where the lexical function model performed the best, with lowest reported median ranks (Q2) above 100.

6 Conclusions

Twelve composition models were evaluated on the task of building compositional representations for German compounds. The best results (median rank 6) were obtained by the newly introduced *Wmask* model, $p = g(W[u \odot u'; v \odot v''])$. The results show that it is possible to learn a composition function specific to compounds, an idea which we would like to further explore using existing compound datasets for English (Ó Séaghdha, 2008; Tratz and Hovy, 2010). The implementation of the newly introduced composition methods can be downloaded from the author's website.

Acknowledgments

The author would like to thank Emanuel Dima, Erhard Hinrichs, Daniël de Kok, Dörte de Kok and Jianqiang Ma, as well as the anonymous reviewers for their insightful comments and 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), project A3.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the Conference on Empirical*

Methods in Natural Language Processing (EMNLP 2010), pages 1183–1193.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the components of German nominal compounds. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI)*, pages 470–474.

Laurie Bauer. 2001. Compounding. In Martin Haspelmath, editor, *Language Typology and Language Universals*. Mouton de Gruyter, The Hague.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011a. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011b. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Georgiana Dinu, The Pham Nghia, and Marco Baroni. 2013a. DISSECT - DIStributional SEmantics Composition Toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 31–36, Sofia, Bulgaria.

Georgiana Dinu, The Pham Nghia, and Marco Baroni. 2013b. General estimation and evaluation of compositional distributional semantic models. In *Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Verena Henrich and Erhard W. Hinrichs. 2011. Determining Immediate Constituents of Compounds in GermaNet. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2011)*, pages 420–426, Hissar, Bulgaria.

Max Kisselew, Sebastian Padó, Alexis Palmer, and Jan Šnajder. 2015. Obtaining a Better Understanding of Distributional Models of German Derivational Morphology. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, pages 58–63, London, UK.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly Derived Representations of Morphologically Complex Words in Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1517–1526, Sofia, Bulgaria.

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.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12.

Lutz Prechelt. 1998. Early stopping - but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Challenges in the Management of Large Corpora (CMLC-3)*.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Stephen Tratz and Eduard 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.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating Linear Models for Compositional Distributional Semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271.