Olive Oil is Made *of* Olives, Baby Oil is Made *for* Babies: Interpreting Noun Compounds using Paraphrases in a Neural Model

Vered Shwartz*
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com

Chris Waterson
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
waterson@google.com

Abstract

Automatic interpretation of the relation between the constituents of a noun compound, e.g. *olive oil* (source) and *baby oil* (purpose) is an important task for many NLP applications. Recent approaches are typically based on either noun-compound representations or paraphrases. While the former has initially shown promising results, recent work suggests that the success stems from memorizing single prototypical words for each relation. We explore a neural paraphrasing approach that demonstrates superior performance when such memorization is not possible.

1 Introduction

Automatic classification of a noun-compound (NC) to the implicit semantic relation that holds between its constituent words is beneficial for applications that require text understanding. For instance, a personal assistant asked "do I have a *morning meeting* tomorrow?" should search the calendar for meetings occurring in the morning, while for *group meeting* it should look for meetings with specific participants. The NC classification task is a challenging one, as the meaning of an NC is often not easily derivable from the meaning of its constituent words (Spärck Jones, 1983).

Previous work on the task falls into two main approaches. The first maps NCs to paraphrases that express the relation between the constituent words (e.g. Nakov and Hearst, 2006; Nulty and Costello, 2013), such as mapping *coffee cup* and *garbage dump* to the pattern $[w_1]$ CONTAINS $[w_2]$. The second approach computes a representation for NCs from the distributional representation of their individual constituents. While this approach yielded promising results, recently, Dima (2016) showed that similar performance is achieved by representing the NC as a concatenation of its constituent embeddings, and attributed it to the *lexical memorization* phenomenon (Levy et al., 2015).

In this paper we apply lessons learned from the parallel task of semantic relation classification. We adapt HypeNET (Shwartz et al., 2016) to the NC classification task, using their path embeddings to represent paraphrases and combining with distributional information. We experiment with various evaluation settings, including settings that make lexical memorization impossible. In these settings, the integrated method performs better than the baselines. Even so, the performance is mediocre for all methods, suggesting that the task is difficult and warrants further investigation.[1]

2 Background

Various tasks have been suggested to address noun-compound interpretation. NC paraphrasing extracts texts explicitly describing the implicit relation between the constituents, for example *student protest* is a protest LED BY, BE SPONSORED BY, or BE ORGANIZED BY students (e.g. Nakov and Hearst, 2006; Kim and Nakov, 2011; Hendrickx et al., 2013; Nulty and Costello, 2013). Compositionality prediction determines to what extent the meaning of the NC can be expressed in terms of the meaning of its constituents, e.g. *spelling bee* is non-compositional, as it is not related to *bee* (e.g. Reddy et al., 2011). In this paper we focus on the NC classification task, which is defined as follows: given a pre-defined set of relations, classify $nc = w_1w_2$ to the relation that holds between w_1 and w_2. We review the various

*Work done during an internship at Google.

[1]The code is available at https://github.com/tensorflow/models/tree/master/research/lexnet_nc.

Proceedings of NAACL-HLT 2018, pages 218–224
New Orleans, Louisiana, June 1 - 6, 2018.

features used in the literature for classification.[2]

2.1 Compositional Representations

In this approach, classification is based on a vector representing the NC (w_1 w_2), which is obtained by applying a function to its constituents' distributional representations: $\vec{v}_{w_1}, \vec{v}_{w_2} \in \mathcal{R}^n$. Various functions have been proposed in the literature.

Mitchell and Lapata (2010) proposed 3 simple combinations of $\vec{v}_{w_1}$ and $\vec{v}_{w_2}$ (additive, multiplicative, dilation). Others suggested to represent compositions by applying linear functions, encoded as matrices, over word vectors. Baroni and Zamparelli (2010) focused on adjective-noun compositions (AN) and represented adjectives as matrices, nouns as vectors, and ANs as their multiplication. Matrices were learned with the objective of minimizing the distance between the learned vector and the observed vector (computed from corpus occurrences) of each AN. The full-additive model (Zanzotto et al., 2010; Dinu et al., 2013) is a similar approach that works on any two-word composition, multiplying each word by a square matrix: $nc = A \cdot \vec{v}_{w_1} + B \cdot \vec{v}_{w_2}$.

Socher et al. (2012) suggested a non-linear composition model. A recursive neural network operates bottom-up on the output of a constituency parser to represent variable-length phrases. Each constituent is represented by a vector that captures its meaning and a matrix that captures how it modifies the meaning of constituents that it combines with. For a binary NC, $nc = g(W \cdot [\vec{v}_{w_1}; \vec{v}_{w_2}])$, where $W \in \mathcal{R}^{2n \times n}$ and g is a non-linear function.

These representations were used as features in NC classification, often achieving promising results (e.g. Van de Cruys et al., 2013; Dima and Hinrichs, 2015). However, Dima (2016) recently showed that similar performance is achieved by representing the NC as a concatenation of its constituent embeddings, and argued that it stems from memorizing prototypical words for each relation. For example, classifying any NC with the head *oil* to the SOURCE relation, regardless of the modifier.

2.2 Paraphrasing

In this approach, the paraphrases of an NC, i.e. the patterns connecting the joint occurrences of the constituents in a corpus, are treated as features. For example, both *paper cup* and *steel knife* may share the feature MADE OF. Séaghdha and Copestake (2013) leveraged this "relational similarity" in a kernel-based classification approach. They combined the relational information with the complementary lexical features of each constituent separately. Two NCs labeled to the same relation may consist of similar constituents (*paper-steel*, *cup-knife*) and may also appear with similar paraphrases. Combining the two information sources has shown to be beneficial, but it was also noted that the relational information suffered from data sparsity: many NCs had very few paraphrases, and paraphrase similarity was based on ngram overlap.

Recently, Surtani and Paul (2015) suggested to represent NCs in a vector space model (VSM) using paraphrases as features. These vectors were used to classify new NCs based on the nearest neighbor in the VSM. However, the model was only tested on a small dataset and performed similarly to previous methods.

3 Model

We similarly investigate the use of paraphrasing for NC relation classification. To generate a signal for the joint occurrences of w_1 and w_2, we follow the approach used by HypeNET (Shwartz et al., 2016). For an $w_1 w_2$ in the dataset, we collect all the dependency paths that connect w_1 and w_2 in the corpus, and learn path embeddings as detailed in Section 3.2. Section 3.1 describes the classification models with which we experimented.

3.1 Classification Models

Figure 1 provides an overview of the models: path-based, integrated, and integrated-NC, each which incrementally adds new features not present in the previous model. In the following sections, $\vec{x}$ denotes the input vector representing the NC. The network classifies NC to the highest scoring relation: $r = \text{argmax}_i \, \text{softmax}(\vec{o})_i$, where $\vec{o}$ is the output layer. All networks contain a single hidden layer whose dimension is $\frac{|x|}{2}$. k is the number of relations in the dataset. See Appendix A for additional technical details.

Path-based. Classifies the NC based only on the paths connecting the joint occurrences of w_1 and w_2 in the corpus, denoted $P(w_1, w_2)$. We define the feature vector as the average of its path embeddings, where the path embedding $\vec{p}$ of a path p is

[2] Leaving out features derived from lexical resources (e.g. Nastase and Szpakowicz, 2003; Tratz and Hovy, 2010).

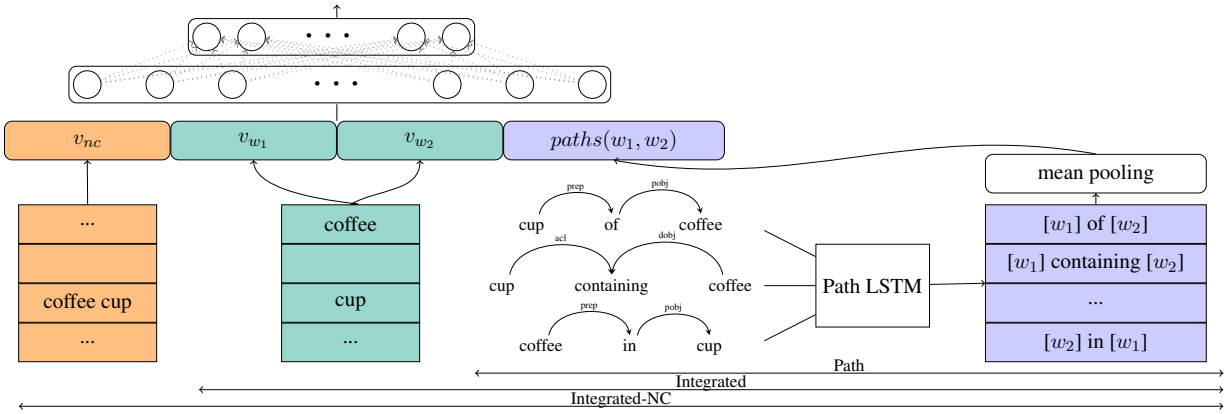

Figure 1: An illustration of the classification models for the NC *coffee cup*. The model consists of two parts: (1) the distributional representations of the NC (left, orange) and each word (middle, green). (2) the corpus occurrences of *coffee* and *cup*, in the form of dependency path embeddings (right, purple).

weighted by its frequency $f_{p,(w_1,w_2)}$:

$$\vec{x} = \vec{v}_{P(w_1,w_2)} = \frac{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \vec{p}}{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)}}$$

Integrated. We concatenate w_1 and w_2's word embeddings to the path vector, to add distributional information: $x = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{P(w_1,w_2)}]$. Potentially, this allows the network to utilize the contextual properties of each individual constituent, e.g. assigning high probability to SUBSTANCE-MATERIAL-INGREDIENT for edible w_1s (e.g. *vanilla pudding, apple cake*).

Integrated-NC. We add the NC's *observed* vector $\vec{v}_{nc}$ as additional distributional input, providing the contexts in which $w_1 w_2$ occur as an NC: $\vec{v}_{nc} = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{nc}, \vec{v}_{P(w_1,w_2)}]$. Like Dima (2016), we learn NC vectors using the GloVe algorithm (Pennington et al., 2014), by replacing each NC occurrence in the corpus with a single token.

This information can potentially help clustering NCs that appear in similar contexts despite having low pairwise similarity scores between their constituents. For example, *gun violence* and *abortion rights* belong to the TOPIC relation and may appear in similar news-related contexts, while *(gun, abortion)* and *(violence, rights)* are dissimilar.

3.2 Path Embeddings

Following HypeNET, for a path p composed of edges $e_1, ..., e_k$, we represent each edge by the concatenation of its lemma, part-of-speech tag, dependency label and direction vectors: $\vec{v_e} = [\vec{v_l}, \vec{v_{pos}}, \vec{v_{dep}}, \vec{v_{dir}}]$. The edge vectors $\vec{v_{e_1}}, ..., \vec{v_{e_k}}$ are encoded using an LSTM (Hochreiter and Schmidhuber, 1997), and the last output vector $\vec{p}$ is used as the path embedding.

We use the NC labels as distant supervision. While HypeNET predicts a word pair's label from the frequency-weighted average of the path vectors, we differ from it slightly and compute the label from the frequency-weighted average of the predictions obtained from each path separately:

$$\vec{o} = \frac{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \text{softmax}(\vec{p})}{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)}}$$

$$r = \text{argmax}_i \, \vec{o}_i$$

We conjecture that label distribution averaging allows for more efficient training of path embeddings when a single NC contains multiple paths.

4 Evaluation

4.1 Dataset

We follow Dima (2016) and evaluate on the Tratz (2011) dataset, with 19,158 instances and two levels of labels: fine-grained (`Tratz-fine`, 37 relations) and coarse-grained (`Tratz-coarse`, 12 relations). We report results on both versions. See Tratz (2011) for the list of relations.

Dataset Splits Dima (2016) showed that a classifier based only on v_{w_1} and v_{w_2} performs on par with compound representations, and that the success comes from lexical memorization (Levy et al., 2015): memorizing the majority label of single words in particular slots of the compound (e.g. TOPIC for *travel guide*, *fishing guide*, etc.). This memorization paints a skewed picture of the state-of-the-art performance on this difficult task.

To better test this hypothesis, we evaluate on 4 different splits of the datasets to train, test, and validation sets: (1) **random**, in a 75:20:5 ratio, (2) **lexical-full**, in which the train, test, and validation

Dataset	Split	Best Freq	Dist	Dist-NC	Best Comp	Path	Int	Int-NC
Tratz-fine	Rand	0.319	0.692	0.673	**0.725**	0.538	0.714	0.692
	Lex_{head}	0.222	0.458	0.449	0.450	0.448	**0.510**	0.478
	Lex_{mod}	0.292	0.574	0.559	0.607	0.472	**0.613**	0.600
	Lex_{full}	0.066	0.363	0.360	0.334	0.423	0.421	**0.429**
Tratz-coarse	Rand	0.256	0.734	0.718	**0.775**	0.586	0.736	0.712
	Lex_{head}	0.225	0.501	0.497	0.538	0.518	**0.558**	0.548
	Lex_{mod}	0.282	0.630	0.600	0.645	0.548	**0.646**	0.632
	Lex_{full}	0.136	0.406	0.409	0.372	0.472	0.475	**0.478**

Table 1: All methods' performance (F_1) on the various splits: **best freq**: best performing frequency baseline (head / modifier),[3] **best comp**: best model from Dima (2016).

Dataset	Split	Train	Validation	Test
TRATZ-FINE	Lex_{full}	4,730	1,614	869
	Lex_{head}	9,185	5,819	4,154
	Lex_{mod}	9,783	5,400	3,975
	Rand	14,369	958	3,831
TRATZ-COARSE	Lex_{full}	4,746	1,619	779
	Lex_{head}	9,214	5,613	3,964
	Lex_{mod}	9,732	5,402	3,657
	Rand	14,093	940	3,758

Table 2: Number of instances in each dataset split.

sets each consists of a distinct vocabulary. The split was suggested by Levy et al. (2015), and it randomly assigns words to distinct sets, such that for example, including *travel guide* in the train set promises that *fishing guide* would not be included in the test set, and the models do not benefit from memorizing that the head *guide* is always annotated as TOPIC. Given that the split discards many NCs, we experimented with two additional splits: (3) **lexical-mod** split, in which the w_1 words are unique in each set, and (4) **lexical-head** split, in which the w_2 words are unique in each set. Table 2 displays the sizes of each split.

4.2 Baselines

Frequency Baselines. *mod freq* classifies w_1w_2 to the most common relation in the train set for NCs with the same modifier (w_1w_2'), while *head freq* considers NCs with the same head ($w_1'w_2$).[4]

Distributional Baselines. Ablation of the path-based component from our models: **Dist** uses only w_1 and w_2's word embeddings: $\vec{x} = [\vec{v}_{w_1}, \vec{v}_{w_2}]$, while **Dist-NC** includes also the NC embedding: $\vec{x} = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{nc}]$. The network architecture is defined similarly to our models (Section 3.1).

Compositional Baselines. We re-train Dima's (2016) models, various combinations of NC representations (Zanzotto et al., 2010; Socher et al., 2012) and single word embeddings in a fully connected network.[5]

4.3 Results

Table 1 shows the performance of various methods on the datasets. Dima's (2016) compositional models perform best among the baselines, and on the random split, better than all the methods. On the lexical splits, however, the baselines exhibit a dramatic drop in performance, and are outperformed by our methods. The gap is larger in the lexical-full split. Finally, there is usually no gain from the added NC vector in Dist-NC and Integrated-NC.

5 Analysis

Path Embeddings. To focus on the changes from previous work, we analyze the performance of the path-based model on the Tratz-fine random split. This dataset contains 37 relations and the model performance varies across them. Some relations, such as MEASURE and PERSONAL_TITLE yield reasonable performance (F_1 score of 0.87 and 0.68). Table 3 focuses on these relations and illustrates the indicative paths that the model has learned for each relation. We compute these by performing the analysis in Shwartz et al. (2016), where each path is fed into the path-based model, and is assigned to its best-scoring relation. For each relation, we consider paths with a score ≥ 0.8.

Other relations achieve very low F_1 scores, indicating that the model is unable to learn them at all. Interestingly, the four relations with the lowest performance in our model[6] are also those with the highest error rate in Dima (2016), very

[3] In practice, in lexical-full this is a random baseline, in lexical-head it is the modifier frequency baseline, and in lexical-mod it is the head frequency baseline.

[4] Unseen heads/modifiers are assigned a random relation.

[5] We only include the compositional models, and omit the "basic" setting which is similar to our Dist model. For the full details of the compositional models, see Dima (2016).

[6] LEXICALIZED, TOPIC_OF_COGNITION&EMOTION, WHOLE+ATTRIBUTE&FEAT, PARTIAL_ATTR_TRANSFER

Relation	Path	Examples
MEASURE	[w_2] varies by [w_1] 2,560 [w_1] portion of [w_2]	*state limit, age limit* *acre estate*
PERSONAL TITLE	[w_2] Anderson [w_1] [w_2] Sheridan [w_1]	*Mrs. Brown* *Gen. Johnson*
CREATE-PROVIDE-GENERATE-SELL	[w_2] produce [w_1] [w_2] selling [w_1] [w_2] manufacture [w_1]	*food producer, drug group* *phone company, merchandise store* *engine plant, sugar company*
TIME-OF1	[w_2] begin [w_1] [w_2] held Saturday [w_1]	*morning program* *afternoon meeting, morning session*
SUBSTANCE-MATERIAL-INGREDIENT	[w_2] made of wood and [w_1] [w_2] material includes type of [w_1]	*marble table, vinyl siding* *steel pipe*

Table 3: Indicative paths for selected relations, along with NC examples.

Test NC		Most Similar NC	
NC	**Label**	**NC**	**Label**
majority party	EQUATIVE	*minority party*	WHOLE+PART_OR_MEMBER_OF
enforcement director	OBJECTIVE	*enforcement chief*	PERFORM&ENGAGE_IN
fire investigator	OBJECTIVE	*fire marshal*	ORGANIZE&SUPERVISE&AUTHORITY
stabilization plan	OBJECTIVE	*stabilization program*	PERFORM&ENGAGE_IN
investor sentiment	EXPERIENCER-OF-EXPERIENCE	*market sentiment*	TOPIC_OF_COGNITION&EMOTION
alliance member	WHOLE+PART_OR_MEMBER_OF	*alliance leader*	OBJECTIVE

Table 4: Example of NCs from the `Tratz-fine` random split test set, along with the most similar NC in the embeddings, where the two NCs have different labels.

likely since they express complex relations. For example, the LEXICALIZED relation contains non-compositional NCs (*soap opera*) or lexical items whose meanings departed from the combination of the constituent meanings. It is expected that there are no paths that indicate lexicalization. In PARTIAL_ATTRIBUTE_TRANSFER (*bullet train*), w_1 transfers an attribute to w_2 (e.g. *bullet* transfers speed to *train*). These relations are not expected to be expressed in text, unless the text aims to explain them (e.g. *train as fast as a bullet*).

Looking closer at the model confusions shows that it often defaulted to general relations like OBJECTIVE (*recovery plan*) or RELATIONAL-NOUN-COMPLEMENT (*eye shape*). The latter is described as "indicating the complement of a relational noun (e.g., son of, price of)", and the indicative paths for this relation indeed contain many variants of "[w_2] of [w_1]", which potentially can occur with NCs in other relations. The model also confused between relations with subtle differences, such as the different topic relations. Given that these relations were conflated to a single relation in the inter-annotator agreement computation in Tratz and Hovy (2010), we can conjecture that even humans find it difficult to distinguish between them.

NC Embeddings. To understand why the NC embeddings did not contribute to the classification, we looked into the embeddings of the `Tratz-fine` test NCs; 3091/3831 (81%) of them had embeddings. For each NC, we looked for the 10 most similar NC vectors (in terms of cosine similarity), and compared their labels. We have found that only 27.61% of the NCs were mostly similar to NCs with the same label. The problem seems to be inconsistency of annotations rather than low embeddings quality. Table 4 displays some examples of NCs from the test set, along with their most similar NC in the embeddings, where the two NCs have different labels.

6 Conclusion

We used an existing neural dependency path representation to represent noun-compound paraphrases, and along with distributional information applied it to the NC classification task. Following previous work, that suggested that distributional methods succeed due to lexical memorization, we show that when lexical memorization is not possible, the performance of all methods is much worse. Adding the path-based component helps mitigate this issue and increase performance.

Acknowledgments

We would like to thank Marius Pasca, Susanne Riehemann, Colin Evans, Octavian Ganea, and Xiang Li for the fruitful conversations, and Corina Dima for her help in running the compositional baselines.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1183–1193.

Corina Dima. 2016. *Proceedings of the 1st Workshop on Representation Learning for NLP*, Association for Computational Linguistics, chapter On the Compositionality and Semantic Interpretation of English Noun Compounds, pages 27–39. https://doi.org/10.18653/v1/W16-1604.

Corina Dima and Erhard Hinrichs. 2015. Automatic noun compound interpretation using deep neural networks and word embeddings. *IWCS 2015* page 173.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Sofia, Bulgaria, pages 50–58. http://www.aclweb.org/anthology/W13-3206.

Iris Hendrickx, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, pages 138–143. http://aclweb.org/anthology/S13-2025.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Nam Su Kim and Preslav Nakov. 2011. Large-scale noun compound interpretation using bootstrapping and the web as a corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 648–658. http://aclweb.org/anthology/D11-1060.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 970–976. http://www.aclweb.org/anthology/N15-1098.

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

Preslav Nakov and Marti Hearst. 2006. Using verbs to characterize noun-noun relations. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, pages 233–244.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Fifth international workshop on computational semantics (IWCS-5)*. pages 285–301.

Paul Nulty and Fintan Costello. 2013. General and specific paraphrases of semantic relations between nouns. *Natural Language Engineering* 19(03):357–384.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 210–218. http://www.aclweb.org/anthology/I11-1024.

Diarmuid O Séaghdha and Ann Copestake. 2013. Interpreting compound nouns with kernel methods. *Natural Language Engineering* 19(3):331–356.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2389–2398. http://www.aclweb.org/anthology/P16-1226.

Richard Socher, Brody Huval, D. Christopher Manning, and Y. Andrew 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. Association for Computational Linguistics, pages 1201–1211. http://aclweb.org/anthology/D12-1110.

Karen Spärck Jones. 1983. Compound noun interpretation problems. Technical report, University of Cambridge, Computer Laboratory.

Nitesh Surtani and Soma Paul. 2015. A vsm-based statistical model for the semantic relation interpretation of noun-modifier pairs. In *RANLP*. pages 636–645.

Stephen Tratz. 2011. *Semantically-enriched parsing for natural language understanding*. University of Southern California.

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*. Association for Computational Linguistics, Uppsala, Sweden, pages 678–687. http://www.aclweb.org/anthology/P10-1070.

Tim Van de Cruys, Stergos Afantenos, and Philippe Muller. 2013. Melodi: A supervised distributional approach for free paraphrasing of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 144–147. http://www.aclweb.org/anthology/S13-2026.

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*. Association for Computational Linguistics, pages 1263–1271.

A Technical Details

To extract paths, we use a concatenation of English Wikipedia and the Gigaword corpus.[7] We consider sentences with up to 32 words and dependency paths with up to 8 edges, including satellites, and keep only 1,000 paths for each noun-compound. We compute the path embeddings in advance for all the paths connecting NCs in the dataset (§3.2), and then treat them as fixed embeddings during classification (§3.1).

We use TensorFlow (Abadi et al., 2016) to train the models, fixing the values of the hyper-parameters after performing preliminary experiments on the validation set. We set the mini-batch size to 10, use Adam optimizer (Kingma and Ba, 2014) with the default learning rate, and apply word dropout with probability 0.1. We train up to 30 epochs with early stopping, stopping the training when the F_1 score on the validation set drops 8 points below the best performing score.

We initialize the distributional embeddings with the 300-dimensional pre-trained GloVe embeddings (Pennington et al., 2014) and the lemma embeddings (for the path-based component) with the 50-dimensional ones. Unlike HypeNET, we do not update the embeddings during training. The lemma, POS, and direction embeddings are initialized randomly and updated during training. NC embeddings are learned using a concatenation of Wikipedia and Gigaword. Similarly to the original GloVe implementation, we only keep the most frequent 400,000 vocabulary terms, which means that roughly 20% of the noun-compounds do not have vectors and are initialized randomly in the model.

[7] https://catalog.ldc.upenn.edu/ldc2003t05