Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions

Majid Yazdani
Computer Science Department
University of Geneva
majid.yazdani@unige.ch

Meghdad Farahmand
Computer Science Department
University of Geneva
meghdad.farahmand@unige.ch

James Henderson
Xerox Research Center Europe
james.henderson@xrce.xerox.com

Abstract

Non-compositionality of multiword expressions is an intriguing problem that can be the source of error in a variety of NLP tasks such as language generation, machine translation and word sense disambiguation. We present methods of non-compositionality detection for English noun compounds using the unsupervised learning of a semantic composition function. Compounds which are not well modeled by the learned semantic composition function are considered non-compositional. We explore a range of distributional vector-space models for semantic composition, empirically evaluate these models, and propose additional methods which improve results further. We show that a complex function such as polynomial projection can learn semantic composition and identify non-compositionality in an unsupervised way, beating all other baselines ranging from simple to complex. We show that enforcing sparsity is a useful regularizer in learning complex composition functions. We show further improvements by training a decomposition function in addition to the composition function. Finally, we propose an EM algorithm over latent compositionality annotations that also improves the performance.

1 Introduction

Multiword Expressions (MWEs) are sequences of words that exhibit some kind of idiosyncrasy. This idiosyncrasy can be semantic, statistical, or syntactic[1]. *Ivory tower*, *speed limit*, and *at large* are examples of semantically, statistically and syntactically idiosyncratic MWEs respectively. Note that an MWE can be idiosyncratic at several levels. In general, semantically idiosyncratic MWEs are commonly referred to as non-compositional (Baldwin and Kim, 2010) and statistically idiosyncratic MWEs are commonly referred to as collocations (Sag et al., 2002). Non-compositional MWEs are those whose meaning can not be readily inferred from the meaning of their constituents and collocations are those MWEs whose constituents co-occur more than expected by chance. Collocations constitute the largest subset of all kinds of MWEs, however, non-compositional ones cause more problems in various NLP tasks, for example word sense disambiguation (McCarthy et al., 2003) and machine translation (Lin, 1999). It may also be more challenging to model non-compositionality than collocational weight as the former has to do with modelling the semantics and the latter can to some extent be modeled by conventional statistical measures such as mutual information. Detecting non-compositionality in an automatic fashion has been the aim of much previous research.

In this paper, we capture non-compositionality of English Noun Compounds (NCs)[2] based on the assumption that the majority of the compounds are compositional, for which a composition function can be learned. This implies that the compounds for which a composition function cannot be learned with a relatively low error are non-compositional.

In previous work on vector-space models of distributional semantics, semantic composition has been commonly assumed to be a trivial predetermined function such as addition, multiplication,

[1] MWEs can have other less significant kinds of idiosyncrasies. For instance lexical idiosyncrasy as in *ad hoc*, and pragmatic idiosyncrasy as in *good morning* (Baldwin and Kim, 2010)

[2] MWEs have various syntactic categories such as noun compounds, verb particle constructions, light verb constructions, etc., with noun compounds and verb particle constructions constituting the most prominent categories of MWEs.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1733–1742, Lisbon, Portugal, 17-21 September 2015. ©2015 Association for Computational Linguistics.

and their weighted variations (Mitchell and Lapata, 2008; Reddy et al., 2011; Salehi et al., 2015). Nevertheless there is some work that regards composition as a more complex function. For instance Widdows (2008) who propose (but doesn't empirically test) the use of Tensor and Convolution products for modelling non-compositionality, Baroni and Zamparelli (2010) who regard adjectives in adjectival-noun compositions as matrices that can be learned by linear regression, and Socher et al. (2012) who present a model that learns phrase composition by means of a recursive neural network. The two latter works show that complex composition models significantly outperform additive and multiplicative functions. In this work, we too assume that composition is arguably a complex function. We believe simplified composition functions, such as additive and multiplicative functions and their weighted variations, while having advantages such as being impervious to overfitting, can not completely capture semantic composition. Nevertheless modelling composition by means of a powerful function can be equally inadequate for our purposes. An overly powerful composition function memorizes all compositional and non-compositional compounds, resulting in overfitting and low learning error that hinders discrimination between compositional and non-compositional compounds. We examine various classes of composition functions, ranging from the least to the most powerful (in terms of learning capacity). We show that complex functions clearly do a better job in modelling semantic composition and in detecting non-compositionality compared to commonly used additive and multiplicative functions.

Compositional compounds are also decomposable; intuitively, their semantics is the union of the semantics of their components. More formally, conditioned on the vector of the compound, vectors of the component words should be *independently* predictable. This principle, together with the assumption that most of the compounds are compositional, leads to the conclusion that a model of composition should be able to be auto-reconstructive: the composition function that maps component-words' vectors to their compound vector should have an associated decomposition function that independently predicts each of the component-words' vectors from this compound vector. An auto-reconstructive model enables us to exploit more data in order to learn semantic composition and predict compositionality. We show that auto-reconstruction can improve the accuracy of composition functions and improve detecting non-compositionality.

To further improve non-compositionality detection, we propose an EM-like detection algorithm based on hidden compositionality annotations. The best composition is the one that is the best fit on all the data points except the non-compositional ones. Since we don't use annotated data at training time, we assume annotations to be hidden variables and iteratively alternate between optimizing the composition function and optimizing the hidden compositionality annotations. We show that this iterative algorithm increases the accuracy of non-compositionality detection compared to the case when training is done on all examples.

We run our experiments on the data set of Farahmand et al. (2015) who provide a set of English NCs which are annotated with non-compositionality judgments. We show that quadratic regression significantly outperforms additive and multiplicative baselines and all other models in modelling semantic composition and identifying the non-compositional NCs. In short, the contributions of our work are: to empirically evaluate various composition functions ranging from simple to overly complex in order to find the most accurate function; to propose, to the best of our knowledge for the first time, a method of identifying non-compositional phrases as phrases for which a composition function cannot be readily learned; to propose learning decomposability as another criterion to detect non-compositionality; and to examine possible ways of improving the accuracy of the models by means of EM on hidden compositionality annotations.

2 Related Work

To the best of our knowledge, attempts to extract non-compositionality in computational linguistics go back to 1998. Tapanainen et al. (1998) propose a method to identify non-compositional verb-object collocations based on the semantic asymmetry of verb-object relation. They assume that in a verb-object idiomatic expression, the object is a more interesting element in the sense that if the object appears with one (or only a few) verbs in a large corpus, it presumably has an id-

iomatic nature. Lin (1999) argues that the mutual information between the constituents of a non-compositional phrase is significantly different from that of a phrase created by substituting the constituents of that phrase with their similar words. Their evaluation reveals a low precision ($16 - 39\%$) and recall ($14 - 21\%$). In any case this method is not able to discriminate non-compositional MWEs from collocational MWEs as they share the same property of non-substitutability (their constituents cannot be replaced with their synonyms). Baldwin et al. (2003) present a method that decides about the non-compositionality of English NCs and verb particle constructions by using latent semantic analysis to calculate the similarity between a MWE and its components. They argue that a higher similarity indicates a higher degree of compositionality. McCarthy et al. (2003) devise a number of measures based on comparison of the neighbors of phrasal verbs and their corresponding simplex verbs. They evaluate these measures by calculating their correlation with human compositionality judgments on a set of phrasal verbs. They show that some of the measures have significant correlations with human judgments. Venkatapathy and Joshi (2005) present a supervised model that benefits from both collocational and contextual information and ranks the MWE candidates based on their non-compositionality. Katz and Giesbrecht (2006) use distributional semantics and LSA as a model of context similarity to test whether the local context of a MWE can distinguish its idiomatic use from literal use. They further compare the context of a MWE with the context of its components and show that this can be used to decide whether the expression is idiomatic or not. Cook et al. (2007) is a relatively different work where the authors propose a syntactic approach to identify semantic non-compositionality of verb-noun MWEs. McCarthy et al. (2007) use various models of selectional preferences for detecting non-compositional verb-object pairs.

Reddy et al. (2011) employ the additive and multiplicative composition functions presented by Mitchell and Lapata (2008)[3] and several similarity based models to measure the compositionality of MWEs. Similarity based models measure the similarity of a MWE vector and sum/product of its constituents' vectors. Their evaluation (which is carried out on a set of 90 annotated NCs) shows that there is a relatively high correlation (Spearman ρ of between 0.51 and 0.71) between their models' predictions and human judgments on non-compositionality of English NCs, with weighted additive function outperforming all the other models. Kiela and Clark (2013) present a model of detecting non-compositionality based on the hypothesis that the average distance between a phrase vector and its substituted phrase vectors is related to its compositionality. In particular compositional phrases are less similar to their neighbors in semantic space. The distributional vectors representing the semantics of words were created using the standard window method and 50,000 most frequent context words. They show that their model slightly ($+0.014$ and $+0.007$) outperforms their baselines (Venkatapathy and Joshi, 2005; McCarthy et al., 2007).

All of the models mentioned so far are based on conventional[4] or count based vector space representation of the words. More recent works however are based on representation learning of word embeddings. Baroni and Zamparelli (2010) regard adjective as matrices and nouns as real-valued vectors for Italian adjective noun composition. They learn the adjective matrices by linear regression. In this work, however, every adjective is presented by a new matrix which leads to a large number of parameters. Socher et al. (2012) suggest that composition function is a matrix that multiplies on the word vectors, and Mikolov et al. (2013b) present a model of learning non-compositional phrases by calculating a data-driven score for certain frequent phrases (up to size two) and learn them as a whole. Salehi et al. (2015) borrow the word embeddings from (Mikolov et al., 2013a) to model the semantics of words and use several composition functions from (Mitchell and Lapata, 2008; Reddy et al., 2011) to predict the non-compositionality of MWEs. They compare the performance of word embeddings with conventional distributional vector representations and discover the superiority of word embeddings in predicting non-compositionality of MWEs.

[3]Mitchell and Lapata (2008) present an analysis of vector-based additive and multiplicative semantic composition models where each words is represented by its distributional vector. They conclude that multiplicative and combined models do a better job in modelling vector-based semantic composition than other models.

[4]Conventional or count based models of distributional similarity as oppose to word embeddings (Salehi et al., 2015; Baroni et al., 2014).

3 Representation of Words and Compounds

In order to represent words and compounds we use word embeddings, which are a form of vector space models. Vector space models represent the semantics of words and phrases with real valued vectors. Word embeddings have proven to be effective models of semantic representation of words and outperform the count-based models in various NLP tasks (Baroni et al., 2014; Collobert et al., 2011; Collobert and Weston, 2008; Yazdani and Popescu-Belis, 2013; Huang et al., 2012; Mikolov et al., 2013c). They have been successfully applied to semantic composition (Mikolov et al., 2013b) and outperformed the conventional count based contextual models in predicting non-compositionality of MWEs (Salehi et al., 2015).

In this work we use word embeddings of Mikolov et al. (2013a) to represent the semantics of words and compounds. We chose an English Wikipedia dump as our corpus. After filtering HTML tags and noise we POS-tagged the corpus and extracted $\approx 70k$ compounds whose frequency of occurrence was above 50. We learn the embeddings of these compounds as single tokens using the word2vec[5] bag-of-word model. We also learn the embeddings of the compounds of the evaluation set, plus the embeddings of all the compounds' component words. Compounds' sizes are restricted to two (i.e. bigrams) for the sake of simplicity and to respect the evaluation set standards. The compounds and word embeddings are then used as supervised signals to learn a composition function.

4 Supervised Models of Composition on Word Embeddings

After the unsupervised learning of word embeddings and candidate compound embeddings (see section 3), we use these embeddings as supervised signals in order to train our composition functions. The term supervised might be misleading as the models do not have any information about the compositionality of the compounds during the training phase, and in that respect it is unsupervised. To describe the models in a formal way, throughout the paper we use the following notations: d represents the size of embeddings, $\phi(w_i)$ represents embedding of w_i, and $\tilde{\phi}(w_i{-}w_j) = f(\phi(w_i), \phi(w_j))$ represents the learned embedding of bigram $w_i{-}w_j$ by the composition function f. The training error of bigram $w_i{-}w_j$ by f is $e_{ij} = \|\tilde{\phi}(w_i{-}w_j) - \phi(w_i, w_j)\|$, and $\|\|$ is norm 2. The composition functions are described in the following sections.

Given unsupervised embeddings for both words and compounds, a composition model is trained to map the word embeddings to the compound embeddings, with norm 2 error e_{ij} defined above. Then this same error for this same task (norm 2 between predicted and unsupervised compound embeddings) is used to measure non-compositionality. In other words, we learn a composition function (with several models) and identify non-compositional expressions as those for which the error of this composition function is high.

We explore various classes of composition functions of word embeddings, ranging from simple to complex, to find the most effective one. We want a composition function that is powerful enough to learn composition for compositional compounds, but simple enough that it fails to learn composition for non-compositional compounds. To this end, we investigate linear projections, polynomial projections, and neural networks. We try these models with and without sparsity regularisation, which reduces the number of non-zero parameters while otherwise keeping the complexity of the function that can be learned.

4.1 Linear Projection

In this model we assume that the embedding of a bigram is a linear projection of its component words' embeddings.

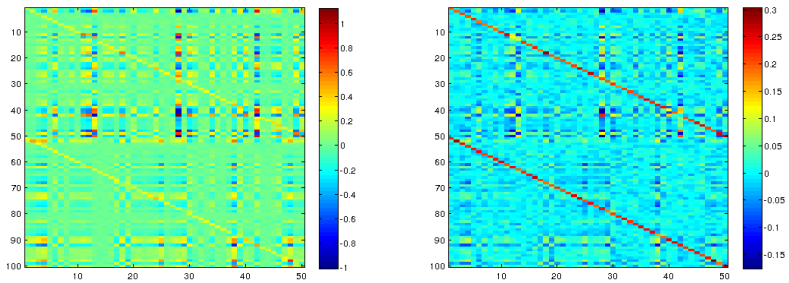
$$f(\phi(w_i), \phi(w_j)) = [\phi(w_i), \phi(w_j)]\theta_{2d\times d}$$

To train this function we optimize the least square error, which gives us a multi-variant linear regression.

$$\min_{\theta} \|[\phi(w_i), \phi(w_j)]\theta_{2d\times d} - \phi(w_i, w_j)\|$$

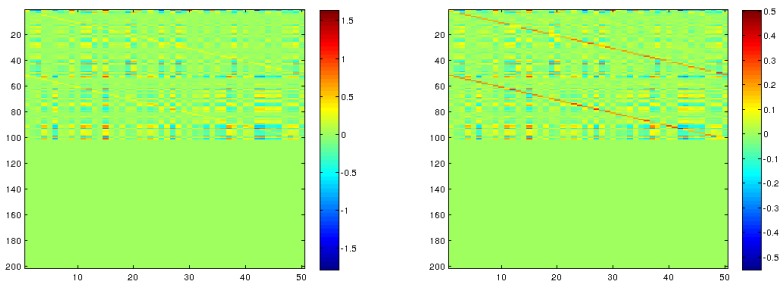
As mentioned earlier, a composition function that doesn't overfit the training data and induces a more meaningful error is more suitable for our purpose. One effective way of reducing overfitting and increasing generalization is by keeping only the important parameters of the model, which is done by enforcing sparsity on model parameters.

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



(a) Linear Projection (b) Sparse Linear Projection

Figure 1: Linear transformation matrix of compositionality for embeddings of size 50



(a) Pure Quadratic (b) Sparse Pure Quadratic

Figure 2: Pure quadratic transformation matrix of compositionality for embeddings of size 50

In case of sparse linear projections, only a few elements of the projection matrix θ are non-zero. This means that not all dimensions of the latent space has a role in all dimensions of the compound embedding.

To apply sparsity on θ, we add a norm 1 penalty on it and add that to the least square optimization. This forms a multi-variant lasso regression (Tibshirani, 2011).

$$\min_{\theta} \|[\phi(w_i), \phi(w_j)]\theta - \phi(w_i, w_j)\| + \lambda|\theta|$$

Figure 1 shows the transformation matrices of linear projection and sparse linear projection. The two diagonals of the matrices correspond to the sum of the two embeddings, which we can see are the main component of the sparse function, and play an important role in the non-sparse one. We will see that despite being an important component of these functions, sum alone is not capable of accurately modelling semantic composition.

4.2 Polynomial Projection

Polynomial projection is a non-linear projection that assumes the relation between compound embedding and the component words' embeddings should be a polynomial of degree n. This can be viewed as a form of linear regression where first a polynomial transformation is applied to the input vector and then a linear projection is fitted. If ψ shows the polynomial transformation then we have:

$$f(\phi(w_i), \phi(w_j)) = \psi([\phi(w_i), \phi(w_j)])\theta$$

We couldn't successfully apply any polynomial beyond quadratic transformation without overfitting. The case of a quadratic ψ transformation is:

$$\psi(x) = \underbrace{x_1^2, \cdots x_n^2}_{\text{Pure quadratic}}, \underbrace{x_1x_2, \cdots x_{n-1}x_n}_{\text{interaction terms}}, \underbrace{x_1, \cdots x_n}_{\text{linear terms}}$$

Similar to the linear case we can have sparse version of the polynomial regression in which we allow the presence of only a few non-zero elements in the θ matrix. The sparsity regularizer is more important in the case of polynomial regression as we have many more parameters. The quadratic model is similar to Recursive Neural Tensor compositionality model of Socher et al. (2013). But in our model the tensor is symmetric around the diagonal. Figure 2 shows the pure quadratic transformation matrices.

4.3 Neural Networks

A feed forward neural network is a universal approximator (Cybenko, 1989): feed-forward network with a single hidden layer can approximate any continuous function, provided it has enough hidden units. Therefore we use neural networks as a powerful class of learning models to learn semantic composition. The number of hidden units gives us a measure to control expressiveness of our model.

$$f(\phi(w_i), \phi(w_j)) = \sigma([\phi(w_i), \phi(w_j)]W_{ih})W_{ho}$$

Similar to the previous models, we optionally impose sparsity over weight matrices of the neural network to be able to induce more meaningful learning errors.

4.4 Experimental Results

We evaluated the above models on the data set of Farahmand et al. (2015). They provide a set of 1042 English NCs with four non-compositionality judgments. The judgments are binary decisions taken by four experts about whether or not a compound is non-compositional. We calculate a vote-based non-compositionality score for each of the data set compounds by summing over its non-compositionality judgments. The neural network models are trained using stochastic gradient descent. We use the additive and multiplicative models of modelling composition and detecting non-compositionality presented by Salehi et al. (2015) and (Reddy et al., 2011) as state of the art baselines.

The results are shown in Table 1. The second column shows the correlation between different models' predictions and the annotated data in terms of Spearman ρ. The last three columns show the performance of different models in terms of Normalized Discounted Cumulative Gain (NDCG), F_1 score and and Precision at 100 ($P@100$). For these three scores we consider the problem of predicting non-compositional NCs a problem with a binary solution where we assume compounds (of the evaluation set) with at least two non-compositionality votes are non-compositional. *NDCG* assigns a higher score to a ranked list of compounds if the non-compositional ones are ranked higher in the list. F_1 column represents the maximum F_1 score on the top-n elements of the ranked list returned by the corresponding model for all n in $[1 - \text{size-of-ranked-list}]$. $P@100$ shows the precision at the first 100 compounds ranked as non-compositional. The models are listed in the order of complexity of the composition function. The addition-based baseline which was explored in a variety of previous work does not seem to be as powerful as the other models. It is outperformed by almost all learned models. In general, we can see that more complex functions tend to learn compositionality in a more effective way.

Model	Spearman ρ	NDCG	F_1	$P@100$
Additive model (Salehi et al., 2015); (Reddy et al., 2011)	20.83	81.39	36.95	43
Multiplicative model (Reddy et al., 2011)	9.18	76	35.61	22
Sparse Linear	37.58	84.25	46.40	48
Linear	38.09	84.25	46.41	49
Sparse Pure Quad.	37.85	84.11	47.05	48
Pure Quad.	38.57	84.68	47.01	47
Sparse Interaction	**41.03**	**85.82**	**48.71**	**54**
Interaction	40.25	85.69	48.64	50
Quadratic	40.25	85.59	48.34	49
Sparse NN (H=1000)	37.08	85.04	46.35	52
NN (H=1000)	37.51	84.97	45.47	51

Table 1: Results for each model's ability to predict non-compositionality.

As mentioned earlier, overly powerful learners overfit and do not produce meaningful errors for the detection task. Sparsity seems to address this issue by reducing the number of non-zero parameters while the function can still keep the complex terms if needed. In general sparse models show improvement over their non-sparse counterparts, specifically for more powerful models.

5 Auto-reconstructive Models

In this section, we investigate the hypothesis that we can detect non-compositionality better by not only modelling a composition function, but also modelling a decomposition function. For compositional compounds, given the meaning of the compound, the meaning of the two component words should be conditionally independent. We therefore assume that the decomposition function predicts the component words' vectors independently. Let us illustrate this assumption by examining the non-compositional compound *flag stop*. Given the semantics of this compound (a point at which a vehicle in public transportation stops only

on prearrangement or signal[6]), we can not readily predict one of its component words without knowing the other. Now consider the compositional compound *hip injury*. Given the semantics of this compound it is much easier to predict each of its component words independently.

In the previous section, the training signals came from the embeddings of the candidate compounds and their component words. In this section we extend our model such that it can benefit from more training signals. To this end, we formalize the assumption that a compositional compound is also decomposable as an auto-reconsructive model. We thus add this hypothesis to the learning process: a good composition function not only builds the semantics of the compound from the semantics of its component words, but it also allows the independent prediction of the semantics of its component words from the compound semantics. In the following sections we add this assumption to both linear projection (which encompasses polynomial) and Neural Network models.

5.1 Auto-reconstructive Linear Models

Let $Y_{M\times d}$ be a matrix whose rows are the pre-computed compound embeddings, and $X_{M\times 2d}$ be a matrix whose rows are the concatenation of the embeddings for the words of these compounds. Let $A_{N\times 2d}$ be another matrix where every row contains the concatenation of the embeddings for the words of a compound, but this matrix includes many compounds for which we did not pre-compute compound embeddings. We assume that the rows of matrix A include the rows of matrix X. In linear models the auto-reconstructive objective function is as follows:

$$\min_{\theta,\theta'} \|X\theta - Y\| + \lambda\|A\theta\theta' - A\|$$

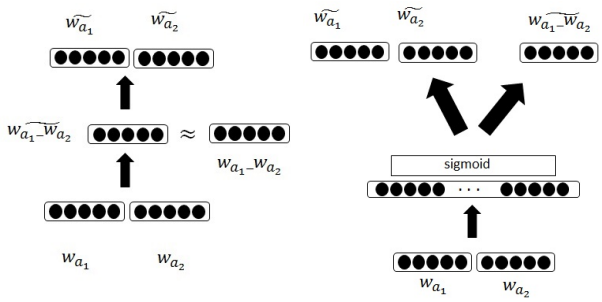
where λ is a meta-parameter for the importance of the auto-reconstruction in the objective. A schematic of this model is shown in Figure 3a.

We can look at this problem as the following weighted least square problem:

$$\min_{\theta,\theta'} \| \begin{pmatrix} X \\ A \end{pmatrix} \theta - \begin{pmatrix} Y \\ A\theta'^T(\theta'\theta'^T)^{-1} \end{pmatrix} \| \begin{pmatrix} 1 \\ \vdots \\ \lambda \end{pmatrix}$$

In the above matrix formula we transformed the auto-reconstruction part of the objective to a pseudo regressand of the least square. To solve this optimization we design an efficient alternating least squares algorithm.

[6]Definition taken from Merriam-Webster Dictionary



(a) Linear auto-reconstructive (b) NN Auto-reconstructive

Figure 3: Auto-reconstructive linear and neural network models

First we initialize θ_0 to be the answer of the original multi-variant linear regression, $\theta_0 = X \setminus Y$ where $X \setminus = (X^T X)^{-1} X^T$ is the pseudoinverse of X. Let us assume W is the diagonal matrix with first M elements of the diagonal being 1 and the remaining N being λ. We alternate between the following formulas until the algorithm converges. First we approximate the next θ' based on the current approximation of θ, then we use this value of θ' to calculate the pseudo regressand part of the least square. In the final step we solve the weighted least square for this new regressand matrix and continue iterating these stages until the algorithm converges.

$$\theta'_t = (A\theta_t) \setminus A \tag{1}$$

$$X_2 = \begin{pmatrix} X \\ A \end{pmatrix}$$

$$Y_2 = \begin{pmatrix} Y \\ A\theta'^T_{t-1}(\theta'_{t-1}\theta'^T_{t-1})^{-1} \end{pmatrix} \tag{2}$$

$$\theta_t = (X_2^T W X_2)^{-1}(X_2)^T W Y_2 \tag{3}$$

The above algorithm can also be used in the case of polynomial regression. The only thing that needs to be done is to replace X and A by their polynomial transformations.

5.2 Auto-reconstructive Neural Networks

The auto-reconstructive neural network follows the same idea. The objective function changes to:

$$\min_{W_{ih},W_{hi},W_{oh}} \|\sigma(XW_{ih})W_{ho} - Y\| + \lambda\|\sigma(AW_{ih})W_{hi} - A\| \tag{4}$$

Composition	Spearman ρ	NDCG	F_1	P@100
Linear	38.09	84.25	46.41	49
Linear+ auto	37.52	84.55	46.55	49
Interaction	**40.25**	85.69	48.64	50
Interaction+ auto	39.29	**85.71**	48.95	**56**
NN (H=1000)	37.40	84.50	46.34	49
NN (H=1000) + auto	39.98	85.17	**49.12**	55

Table 2: Results comparing the auto-reconstructive models' ability to predict non-compositionality.

Composition	Spearman ρ	NDCG	F_1	P@100
Linear	38.09	84.25	46.41	49
Linear+ LA	37.80	84.60	46.29	48
Interaction	40.25	85.69	**48.64**	50
Interaction+ LA	**40.56**	86.30	48.34	51
NN (H=1000)	37.40	84.50	46.34	49
NN (H=1000) + LA	39.23	**85.36**	48.15	**55**

Table 3: Results comparing the latent annotation models' ability to predict non-compositionality.

Figure 3b shows the schematic of this model. We optimize this objective using stochastic gradient descent with early stopping. The results are shown in Table 2. We choose the first $300K$ frequent noun-noun compounds from the corpus in order to build matrix A. Each row of A created by concatenating the component words vectors. The results show that the auto-reconstructive models generally improve over their counterparts. As mentioned earlier, the improvement comes from two facts. On the one hand we increase the training signals by implementing the decomposability hypothesis. On the other hand, the auto-reconstructive model enables us to exploit more data in addition to the candidate compounds. There is almost no improvement in the case of linear model because this model does not have enough learning capacity to benefit from a higher number of training signals.

6 Non-compositionality Detection Using Latent Annotations

All the models discussed in this paper are unsupervised since they don't have any access to labels specifying compositionality of compounds. The above models simply assume that most compounds are compositional, and therefore train their composition and decomposition functions on all compounds. In this section we incorporate in the models an intrinsic uncertainty about the compositionality annotation of the training set.

The best (optimum) composition function is the one that fits well all the compositional compounds and does not fit the non-compositional ones. But we assume that we do not have training labels indicating compositionality. To overcome this uncertainty and improve the learning process, we introduce latent compositionality labels to the model. We assume each candidate compound has a latent annotation, 1 or 0, showing whether or not it is compositional. Let us assume a non-compositionality detection system that returns B non-compositional candidates that should have their own lexical unit and parameters. The objective of this composition function training is to minimize the error of compositional compounds and not the error of non-compositional ones. In order to implement this objective we use the following loss function:

$$\min_{\lambda_{ij},\theta} \sum_{ij} \lambda_{ij} e_{ij}^2$$

$$\text{s.t } \lambda_{ij} \in \{0,1\},$$

$$\sum_{ij} \lambda_{ij} = N - B$$

where λ_{ij} represents the hidden compositionality annotation and e_{ij} is again the learning error for the pair $w_i - w_j$. We want to find the B points such that annotating them as non-compositional results in the minimum error of this objective. The algorithm that alternates between optimizing the composition learning and the hidden annotations eventually converges to this solution.

If the errors are fixed, the B compounds with the biggest errors are the answers to the non-compositional annotation optimization of that iteration. Therefore to solve this optimization we follow an EM-like algorithm: First we set all λ_{ij} to 1 and perform the optimization on the composition function. Then we sort the compounds by their error and set the λ_{ij} of the biggest B elements to 0, and the rest to 1. In other words we assume the compounds with big error are presumably non-compositional according to what we know until that iteration. We continue alternating between training the composition function and annotating high error points until the algorithm reaches convergence. The results are shown in Table 3. Models that use latent annotations clearly outperform their counterparts, especially in terms

of precision at 100. This is expected since at training time we consider a model that returns B non-compositional compounds and therefore precision at 100 is optimized. The latent annotations do not improve the linear model since the model is simple and there is not much room to improve its learning.

7 Conclusions

We proposed a framework to detect non-compositional compounds as the compounds that stand out as outliers in the process of learning compositionality of English noun compounds. We proposed and evaluated a range of functions with a variety of complexities that model semantic composition. We showed that learners such as polynomial projection and neural networks which are distinctly more complex than commonly used additive and multiplicative functions can model semantic composition more effectively. We showed that a function as complex as quadratic projection is a better learner of compositionality than simpler models. We further showed that enforcing sparsity is an effective way of learning a complex composition function while avoiding overfitting and producing meaningful learning errors. Furthermore, we improved our models by incorporating an auto-reconstructive loss function that enables us to benefit from more training signals and cover more data. Finally, we addressed the intrinsic label uncertainty in training data by considering latent annotations, and showed that it can further improve the results.

Acknowledgments

This research was partially funded by Hasler foundation project no. 15019, "Deep Neural Network Dependency Parser for Context-aware Representation Learning".

References

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. *Handbook of Natural Language Processing, second edition. Morgan and Claypool*.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 89–96. Association for Computational Linguistics.

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*, pages 1183–1193. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Ronan Collobert and Jason 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*, ICML '08, pages 160–167, New York, NY, USA. ACM.

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

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the workshop on a broader perspective on multiword expressions*, pages 41–48. Association for Computational Linguistics.

G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Meghdad Farahmand, Aaron Smith, and Joakim Nivre. 2015. A multiword expression data set: Annotating non-compositionality and conventionalization for english noun compounds. In *Proceedings of the 11th Workshop on Multiword Expressions (MWE-NAACL 2015)*. Association for Computational Linguistics.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, MWE '06, pages 12–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douwe Kiela and Stephen Clark. 2013. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *EMNLP*, pages 1427–1432.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 317–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions: Analysis, Acqusation and Treatment*, pages 73–80.

Diana McCarthy, Sriram Venkatapathy, and Aravind K Joshi. 2007. Detecting compositionality of verb-object combinations using selectional preferences. In *EMNLP-CoNLL*, pages 369–379.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *IJCNLP*, pages 210–218.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of NAACL HLT*. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October. Association for Computational Linguistics.

Pasi Tapanainen, Jussi Piitulainen, and Timo Järvinen. 1998. Idiomatic object usage and support verbs. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 1289–1293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Tibshirani. 2011. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Sriram Venkatapathy and Aravind K. Joshi. 2005. Measuring the relative compositionality of verb-noun (v-n) collocations by integrating features. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 899–906, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, volume 26, page 28th.

Majid Yazdani and Andrei Popescu-Belis. 2013. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artif. Intell.*, 194:176–202.