Improving the Lexical Function Composition Model with Pathwise Optimized Elastic-Net Regression

Jiming Li and **Marco Baroni** and **Georgiana Dinu**
Center for Mind/Brain Sciences
University of Trento, Italy
(jiming.li|marco.baroni|georgiana.dinu)@unitn.it

Abstract

In this paper, we show that the lexical function model for composition of distributional semantic vectors can be improved by adopting a more advanced regression technique. We use the pathwise coordinate-descent optimized elastic-net regression method to estimate the composition parameters, and compare the resulting model with several recent alternative approaches in the task of composing simple intransitive sentences, adjective-noun phrases and determiner phrases. Experimental results demonstrate that the lexical function model estimated by elastic-net regression achieves better performance, and it provides good qualitative interpretability through sparsity constraints on model parameters.

1 Introduction

Vector-based distributional semantic models of word meaning have gained increased attention in recent years (Turney and Pantel, 2010). Different from formal semantics, distributional semantics represents word meanings as vectors in a high-dimensional semantic space, where the dimensions are given by co-occurring contextual features. The intuition behind these models lies in the fact that words which are similar in meaning often occur in similar contexts, e.g., *moon* and *star* might both occur with *sky, night* and *bright*. This leads to convenient ways to measure similarity between different words using geometric methods (e.g., the cosine of the angle between two vectors that summarize their contextual distribution). Distributional semantic models have been successfully applied to many tasks in linguistics and cognitive science (Griffiths et al., 2007; Foltz et al., 1998; Laham, 1997; McDonald and Brew, 2004). However, most of these tasks only deal with isolated words, and there is a strong need to construct representations for longer linguistic structures such as phrases and sentences. In order to achieve this goal, the principle of compositionality of linguistic structures, which states that complex linguistic structures can be formed through composition of simple elements, is applied to distributional vectors. Therefore, in recent years, the problem of composition within distributional models has caught many researchers' attention (Clark, 2013; Erk, 2012).

A number of compositional frameworks have been proposed and tested. Mitchell and Lapata (2008) propose a set of simple component-wise operations, such as multiplication and addition. Later, Guevara (2010) and Baroni and Zamparelli (2010) proposed more elaborate methods, in which composition is modeled as matrix-vector multiplication operations. Particularly new to their approach is the proposal to estimate model parameters by minimizing the distance of the composed vectors to corpus-observed phrase vectors. For example, Baroni and Zamparelli (2010) consider the case of Adjective-Noun composition and model it as matrix-vector multiplication: adjective matrices are parameters to be estimated and nouns are co-occurrence vectors. The model parameter estimation procedure becomes a multiple response multivariate regression problem. This method, that, following Dinu et al. (2013) and others, we term the *lexical function* composition model, can also be generalized to more complex structures such as 3rd order tensors for modeling transitive verbs (Grefenstette et al., 2013).

Socher et al. (2012) proposed a more complex and flexible framework based on matrix-vector representations. Each word or lexical node in a parsing tree is assigned a vector (representing inherent meaning of the constituent) and a matrix (controlling the behavior to modify the meaning of

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 434–442,
Gothenburg, Sweden, April 26-30 2014.

Model	Composition function	Parameters
Add	$w_1\vec{u} + w_2\vec{v}$	w_1, w_2
Mult	$\vec{u}^{w_1} \odot \vec{v}^{w_2}$	w_1, w_2
Dil	$\|\|\vec{u}\|\|_2^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$	λ
Fulladd	$W_1\vec{u} + W_2\vec{v}$	$W_1, W_2 \in \mathbf{R}^{m\times m}$
Lexfunc	$A_u\vec{v}$	$A_u \in \mathbf{R}^{m\times m}$
Fulllex	$\tanh([W_1, W_2]\begin{bmatrix} A_u\vec{v} \\ A_v\vec{u} \end{bmatrix})$	$W_1, W_2,$ $A_u, A_v \in \mathbf{R}^{m\times m}$

Table 1: Composition functions of inputs (u, v).

neighbor words or phrases) simultaneously. They use recursive neural networks to learn and construct the entire model and show that it reaches state-of-the-art performance in various evaluation experiments.

In this paper, we focus on the simpler, linear lexical function model proposed by Baroni and Zamparelli (2010) (see also Coecke et al. (2010)) and show that its performance can be further improved through more advanced regression techniques. We use the recently introduced elastic-net regularized linear regression method, which is solved by the pathwise coordinate descent optimization algorithm along a regularization parameter path. This new regression method can rapidly generate a sequence of solutions along the regularization path. Performing cross-validation on this parameter path should yield a much more accurate model for prediction. Besides better prediction accuracy, the elastic-net method also brings interpretability to the composition procedure through sparsity constraints on the model.

The rest of this paper is organized as follows: In Section 2, we give details on the above-mentioned composition models, which will be used for comparison in our experiments. In Section 3, we describe the pathwise optimized elastic-net regression algorithm. Experimental evaluation on three composition tasks is provided in Section 4. In Section 5 we conclude and suggest directions for future work.

2 Composition Models

Mitchell and Lapata (2008; 2010) present a set of simple but effective models in which each component of the output vector is a function of the corresponding components of the inputs. Given input vectors $\vec{u}$ and $\vec{v}$, the weighted additive model (**Add**) returns their weighted sum: $\vec{p} = w_1\vec{u} + w_2\vec{v}$. In the dilation model (**Dil**), the output vector is obtained by decomposing one of the input vectors, say $\vec{v}$, into a vector parallel to $\vec{u}$ and an orthogonal vector, and then dilating only the parallel vector by a factor λ before re-combining (formula in Table 1). Mitchell and Lapata also propose a simple multiplicative model in which the output components are obtained by component-wise multiplication of the corresponding input components. We use its natural *weighted* extension (**Mult**), introduced by Dinu et al. (2013), that takes w_1 and w_2 powers of the components before multiplying, such that each phrase component p_i is given by: $p_i = u_i^{w_1} v_i^{w_2}$.

Guevara (2010) and Zanzotto et al. (2010) explore a full form of the additive model (**Fulladd**), where the two vectors entering a composition process are pre-multiplied by weight matrices before being added, so that each output component is a weighted sum of *all* input components: $\vec{p} = W_1\vec{u} + W_2\vec{v}$.

Baroni and Zamparelli (2010) and Coecke et al. (2010), taking inspiration from formal semantics, characterize composition as *function application*. For example, Baroni and Zamparelli model adjective-noun phrases by treating the adjective as a regression function from nouns onto (modified) nouns. Given that linear functions can be expressed by matrices and their application by matrix-by-vector multiplication, a functor (such as the adjective) is represented by a matrix A_u to be composed with the argument vector $\vec{v}$ (e.g., the noun) by multiplication, returning the *lexical function* (**Lexfunc**) representation of the phrase: $\vec{p} = A_u\vec{v}$.

The method proposed by Socher et al. (2012) can be seen as a combination and non-linear extension of Fulladd and Lexfunc (that Dinu and colleagues thus called **Fulllex**) in which *both* phrase elements act as functors (matrices) *and* arguments (vectors). Given input terms *u* and *v* represented by $(\vec{u}, A_u)$ and $(\vec{v}, A_v)$, respectively, their composition vector is obtained by applying first a linear transformation and then the hyperbolic tangent function to the concatenation of the products $A_u\vec{v}$ and $A_v\vec{u}$ (see Table 1 for the equation). Socher and colleagues also present a way to construct matrix representations for specific phrases, needed to scale this composition method to larger constituents. We ignore it here since we focus on the two-word case.

Parameter estimation of the above composition models follows Dinu et al. (2013) by minimizing the distance to corpus-extracted phrase vectors. In

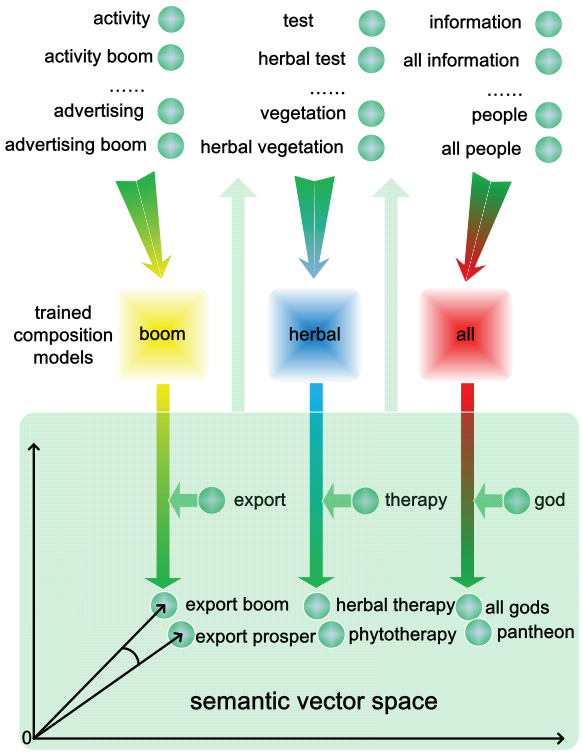

Figure 1: A sketch of the composition model training and composing procedure.

the case of the Fulladd and Lexfunc models this amounts to solving a multiple response multivariate regression problem.

The whole composition model training and phrase composition procedure is described with a sketch in Figure 1. To illustrate with an example, given an intransitive verb *boom*, we want to train a model for this intransitive verb so that we can use it for composition with a noun subject (e.g., *export*) to form an intransitive sentence (e.g., *export boom(s)*). We treat these steps as a composition model learning and predicting procedure. The training dataset is formed with pairs of input (e.g., *activity*) and output (e.g., *activity boom*) vectors. All composition models except Lexfunc also use the functor vector (*boom*) in the training data. Lexfunc does not use this functor vector, but it would rather like to encode the learning target's vector meaning in a different way (see experimental analysis in Section 4.3). Then, this dataset is used for parameter estimation of models. When a model (*boom*) is trained and given a new input semantic vector (e.g., *export*), it will output another vector representing the concept for *export boom*. And the concept *export boom* should be close to similar concepts (e.g., *export prosper*) in meaning under some distance metric in semantic vector space. The same training and composition scheme is applied for other types of functors (e.g., adjectives and determiners). All the above mentioned composition models are evaluated within this scheme, but note that in the case of Add, Dil, Mult and Fulladd, a single set of parameters is obtained across all functors of a certain syntactic category.

3 Pathwise Optimized Elastic-net Algorithm

The elastic-net regression method (Zou and Hastie, 2005) is proposed as a compromise between lasso (Tibshirani, 1996) and ridge regression (Hastie et al., 2009). Suppose there are N observation pairs (x_i, y_i), here $x_i \in \mathbb{R}^p$ is the ith training sample and $y_i \in \mathbb{R}$ is the corresponding response variable in the typical regression setting. For simplicity, assume the x_{ij} are standardized: $\sum_{i=1}^{N} x_{ij}^2 = 1$, for $j = 1, \ldots, p$. The elastic-net solves the following problem:

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \mathbf{P}_\alpha(\beta) \right] \quad (1)$$

where

$$\begin{aligned} \mathbf{P}_\alpha(\beta) &= \lambda((1-\alpha)\frac{1}{2} \parallel \beta \parallel_{\ell_2}^2 + \alpha\beta_{\ell_1}) \\ &= \sum_{j=1}^{p} [\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|]. \end{aligned}$$

$\mathbf{P}$ is the elastic-net penalty, and it is a compromise between the ridge regression penalty and the lasso penalty. The merit of the elastic-net penalty depends on two facts: the first is that elastic-net inherits lasso's characteristic to shrink many of the regression coefficients to zero, a property called sparsity, which results in better interpretability of model; the second is that elastic-net inherits ridge regression's property of a grouping effect, which means important correlated features can be contained in the model simultaneously, and not be omitted as in lasso.

For these linear-type regression problem (ridge, lasso and elastic-net), the determination of the λ value is very important for prediction accuracy. Efron et al. (2004) developed an efficient algorithm to compute the entire regularization path for the lasso problem in 2004. Later, Friedman et al. (Friedman et al., 2007; Friedman et al., 2010) proposed a coordinate descent optimization

method for the regularization parameter path, and they also provided a solution for elastic-net. The main idea of pathwise coordinate descent is to solve the penalized regression problem along an entire path of values for the regularization parameters λ, using the current estimates as warm starts. The idea turns out to be quite efficient for elastic-net regression. The procedure can be described as below: firstly establish an 100 λ value sequence in log scale, and for each of the 100 regularization parameters, use the following coordinate-wise updating rule to cycle around the features for estimating the corresponding regression coefficients until convergence.

$$\tilde{\beta}_j \leftarrow \frac{S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\alpha\right)}{1+\lambda(1-\alpha)} \quad (2)$$

where

- $\tilde{y}_i^{(j)} = \tilde{\beta}_0 + \sum_{\ell \neq j} x_{i\ell}\tilde{\beta}_\ell$ is the fitted value excluding the contribution from x_{ij}, and hence $y_i - \tilde{y}_i^{(j)}$ the partial residual for fitting β_j.
- $S(z, \gamma)$ is the soft-thresholding operator with value

$$\begin{aligned} S(z,\gamma) &= \text{sign}(z)(|z|-\gamma)_+ \\ &= \begin{cases} z-\gamma & \text{if } z>0 \text{ and } \gamma<|z| \\ z+\gamma & \text{if } z<0 \text{ and } \gamma<|z| \\ 0 & \text{if } \gamma \geq |z| \end{cases} \end{aligned}$$

Then solutions for a decreasing sequence of values for λ are computed in this way, starting at the smallest value $\lambda_{\max}$ for which the entire coefficient vector $\hat{\beta} = 0$. Then, 10-fold cross validation on this regularization path is used to determine the best model for prediction accuracy. The α parameter controls the model sparsity (the number of coefficients equal to zero) and grouping effect (shrinking highly correlated features simultaneously).

In what follows, we call the elastic-net regression lexical function model **EnetLex**. In Section 4, we will report the experiment results by EnetLex with $\alpha = 1$. It equals to pathwise coordinate descent optimized lasso, which favours sparser solutions and is often a better estimator when the number of training samples is far greater than the number of feature dimensions, as in our case. We also experimented with intermediate α values (e.g., $\alpha = 0.5$), that were, consistently, inferior or equal to the lasso setting.

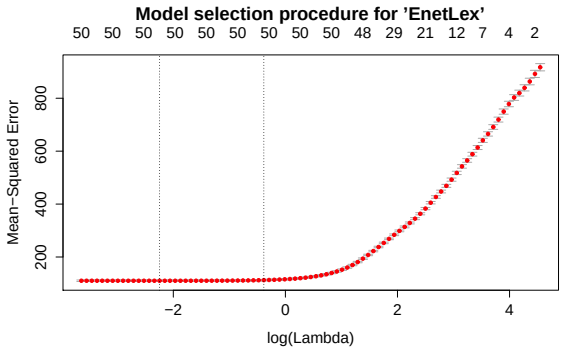

Figure 2: Example of model selection procedure for elastic-net regression (*"the"* model for determiner phrase experiment, SVD, 50 dimensions).

Figure 2 is an example of the model selection procedure between different regularization parameter λ values for determiner *"the"* (experimental details are described in section 4). When α is fixed, EnetLex first generates a λ sequence from $\lambda_{\max}$ to $\lambda_{\min}$ ($\lambda_{\max}$ is set to the smallest value which will shrink all the regression coefficients to zero, $\lambda_{\min} = 0.0001$) in log scale (rightmost point in the plot). The red points corresponding to each λ value in the plot represent mean cross-validated errors and their standard errors. To estimate a model corresponding to some λ value except $\lambda_{\max}$, we use the solution from previous λ value as the initial coefficients (the *warm starts* mentioned before) for iteration with coordinate descent. This will often generate a stable solution path for the whole λ sequence very fast. And we can choose the model with minimum cross-validation error on this path and use it for more accurate prediction. In Figure 2, the labels on the top are numbers of corresponding selected variables (features), the right vertical dotted line is the largest value of lambda such that error is within 1 standard error of the minimum, and the left vertical dotted line corresponds to the λ value which gives minimum cross-validated error. In this case, the λ value of minimum cross-validated error is 0.106, and its log is -2.244316. In all of our experiments, we will select models corresponding to minimum training-data cross-validated error.

4 Experiments

4.1 Datasets

We evaluate on the three data sets described below, that were also used by Dinu et al. (2013), our most

direct point of comparison.

Intransitive sentences The first dataset, introduced by Mitchell and Lapata (2010), focuses on the composition of intransitive verbs and their noun subjects. It contains a total of 120 sentence pairs together with human similarity judgments on a 7-point scale. For example, *value slumps/value declines* is scored 7, *skin glows/skin burns* is scored 1. On average, each pair is rated by 30 participants. Rather than evaluating against mean scores, we use each rating as a separate data point, as done by Mitchell and Lapata. We report Spearman correlations between human-assigned scores and cosines of model-generated vector pairs.

Adjective-noun phrases Turney (2012) introduced a dataset including both noun-noun compounds and adjective-noun phrases (ANs). We focus on the latter, and we frame the task as in Dinu et al. (2013). The dataset contains 620 ANs, each paired with a single-noun paraphrase. Examples include: *upper side/upside*, *false belief/fallacy* and *electric refrigerator/fridge*. We evaluate a model by computing the cosine of all 20K nouns in our semantic space with the target AN, and looking at the rank of the correct paraphrase in this list. The lower the rank, the better the model. We report median rank across the test items.

Determiner phrases The third dataset, introduced in Bernardi et al. (2013), focuses on a class of determiner words. It is a multiple-choice test where target nouns (e.g., *omniscience*) must be matched with the most closely related determiner(-noun) phrases (DPs) (e.g., *all knowledge*). There are 173 target nouns in total, each paired with one correct DP response, as well as 5 foils, namely the determiner (*all*) and noun (*knowledge*) from the correct response and three more DPs, two of which contain the same noun as the correct phrase (*much knowledge, some knowledge*), the third the same determiner (*all preliminaries*). Other examples of targets/related-phrases are *quatrain/four lines* and *apathy/no emotion*. The models compute cosines between target noun and responses and are scored based on their accuracy at ranking the correct phrase first.

4.2 Setup

We use a concatenation of ukWaC, Wikipedia (2009 dump) and BNC as source corpus, totaling 2.8 billion tokens.[1] Word co-occurrences are collected within sentence boundaries (with a maximum of a 50-words window around the target word). Following Dinu et al. (2013), we use the top 10K most frequent content lemmas as context features, Pointwise Mutual Information as weighting method and we reduce the dimensionality of the data by both Non-negative Matrix Factorization (NMF, Lee and Seung (2000)) and Singular Value Decomposition (SVD). For both data dimensionality reduction techniques, we experiment with different numbers of dimension varying from 50 to 300 with a step of 50. Since the Mult model works very poorly when the input vectors contain negative values, as is the case with SVD, for this model we report result distributions across the 6 NMF variations only.

We use the DIStributional SEmantics Composition Toolkit (DISSECT)[2] which provides implementations for all models we use for comparison. Following Dinu and colleagues, we used ordinary least-squares to estimate Fulladd and ridge for Lexfunc. The EnetLex model is implemented in R with support from the glmnet package,[3] which implements pathwise coordinate descent elastic-net regression.

Model	Reduction	Dim	Correlation
Add	NMF	150	0.1349
Dil	NMF	300	0.1288
Mult	NMF	250	0.2246
Fulladd	SVD	300	0.0461
Lexfunc	SVD	250	0.2673
Fulllex	NMF	300	0.2682
EnetLex	SVD	250	**0.3239**

Table 2: Best performance comparison for intransitive verb sentence composition.

4.3 Experimental Results and Analysis

The experimental results are shown in Tables 2, 3, 4 and Figures 3, 4, 5. The best performances from each model on the three composition tasks are shown in the tables. The overall result distributions across reduction techniques and dimensionalities are displayed in the figure

[1] http://wacky.sslmit.unibo.it; http://www.natcorp.ox.ac.uk

[2] http://clic.cimec.unitn.it/composes/toolkit/

[3] http://cran.r-project.org/web/packages/glmnet/

Model	Reduction	Dim	Rank
Add	NMF	300	113
Dil	NMF	300	354.5
Mult	NMF	300	146.5
Fulladd	SVD	300	123
Lexfunc	SVD	150	117.5
Fulllex	SVD	50	394
EnetLex	SVD	300	**108.5**

Table 3: Best performance comparison for adjective noun composition (lower ranks mean better performance).

Model	Reduction	Dim	Rank
Add	NMF	100	0.3237
Dil	NMF	100	0.3584
Mult	NMF	300	0.2023
Fulladd	NMF	200	0.3642
Lexfunc	SVD	200	0.3699
Fulllex	SVD	100	0.3699
EnetLex	SVD	250	**0.4046**

Table 4: Best performance comparison for determiner phrase composition.

boxplots (NMF and SVD results are shown separately). From Tables 2, 3, 4, we can see that EnetLex consistently achieves the best composition performance overall, also outperforming the standard lexical function model. In the boxplot display, we can see that SVD is in general more stable across dimensionalities, yielding smaller variance in the results than NMF. We also observe, more specifically, larger variance in EnetLex performance on NMF than in Lexfunc, especially for determiner phrase composition. The large variance with EnetLex comes from the NMF low-dimensionality results, especially the 50 dimensions condition. The main reason for this lies in the fast-computing tricks of the coordinate descent algorithm when cycling around many features with zero values (as resulting from NMF), which cause fast convergence at the beginning of the regularization path, generating an inaccurate model. A subordinate reason might lie in the unstandardized larger values of the NMF features (causing large gaps between adjacent parameter values in the regularization path). Although data standardization or other feature scaling techniques are often adopted in statistical analysis, they are seldom used in semantic composition tasks due to

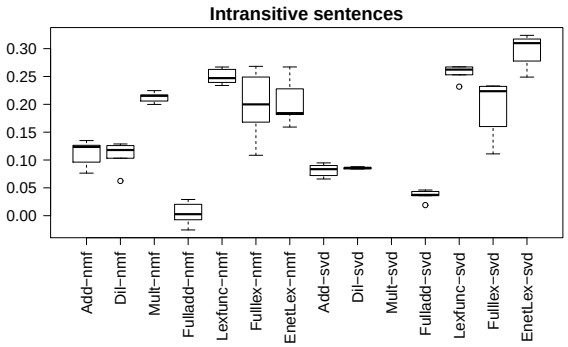

Figure 3: Intransitive verb sentence composition results.

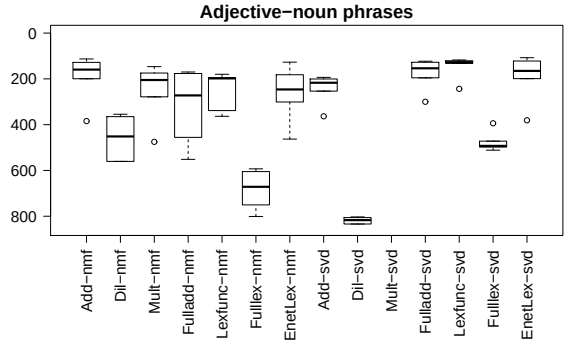

Figure 4: Adjective noun phrase composition results.

the fact that they might negatively affect the semantic vector space. A reasonable way out of this problem would be to save the mean and standard deviation parameters used for data standardization and use them to project the composed phrase vector outputs back to the original vector space.

On the other hand, EnetLex obtained a stable good performance in SVD space, with the best results achieved with dimensions between 200 and 300. A set of Tukey's Honestly Significant Tests show that EnetLex significantly outperforms the other models across SVD settings for determiner phrases and intransitive sentences. The difference is not significant for most comparisons in the adjective phrases task.

For the simpler models for which it was computationally feasible, we repeated the experiments without dimensionality reduction. The results obtained with (unweighted) Add and Mult using full-space representations are reported in Table 5. Due to computational limitations, we tuned full-space weights for Add model only, obtaining similar results to those reported in the table. The full-space

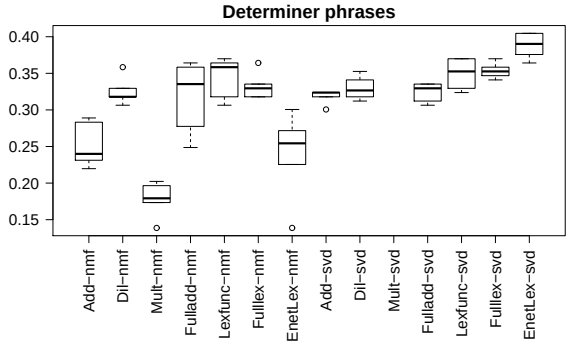

Figure 5: Determiner phrase composition results.

model	verb	adjective	determiner
Add	0.0259	957	0.2832
Mult	0.1796	298.5	0.0405

Table 5: Performance of Add and Mult models without dimensionality reduction.

results confirm that dimensionality reduction is not only a computational necessity when working with more complex models, but it is actually improving the quality of the underlying semantic space.

Another benefit that elastic-net has brought to us is the sparsity in coefficient matrices. Sparsity here means that many entries in the coefficient matrix are shrunk to 0. For the above three experiments, the mean adjective, verb and determiner models' sparsity ratios are 0.66, 0.55 and 0.18 respectively. Sparsity can greatly reduce the space needed to store the lexical function model, especially when we want to use higher orders of representation. Moreover, sparsity in the model is helpful to interpret the concept a specific functor word is conveying. For example, we show how to analyze the coefficient matrices for functor content words (verbs and adjectives). The verb *burst* and adjective *poisonous*, when estimated in the space projected to 100 dimensions with NMF, have percentages of sparsity 47% and 39% respectively, which means 47% of the entries in the *burst* matrix and 39% of the entries in the *poisonous* matrix are zeros.[4] Most of the (hopefully) irrelevant dimensions were discarded during model training. For visualization, we list the 6 most significant columns and rows from verb *burst* and adjective *poisonous* in Table 6. Each reduced NMF dimension is represented by the 3 largest original-context entries in the corresponding row of the NMF basis matrix. The top columns and rows are selected by ordering sums of row entries and sums of column entries (the 10 most common features across trained matrices are omitted). In the matrix-vector multiplication scenario, a larger column contributes more to all the features of the composed output phrase vector, while one large row corresponds to a large composition output feature. From these tables, we can see that the selected top columns and rows are mostly semantically relevant to the corresponding functor words (*burst* and *poisonous*, in the displayed examples).

A very interesting aspect of these experiments is the role of the intercept in our regression model. The path-wise optimization algorithm starts with a lambda value ($\lambda_{\max}$), which sets all the coefficients exactly to 0, and at that time the intercept is just the expected mean value of the training phrase vectors, which in turn is of course quite similar to the co-occurrence vector of the corresponding functor word (by averaging the *poisonous N* context distributions, we obtain a vector that approximates the *poisonous* distribution). And, although the intercept also changes with different lambda values, it still highly correlates with the co-occurrence vectors of the functor words in vector space. For adjectives and verbs, we compared the initial model's ($\lambda_{\max}$) intercept and the minimum cross-validation error model intercept with corpus-extracted vectors for the corresponding words. That is, we used the word co-occurrence vector for a verb or an adjective extracted from the corpus and projected onto the reduced feature space (e.g., NMF, 100 dimensions), then computed cosine similarity between this word meaning representation and its corresponding EnetLex matrix initial and minimum-error intercepts, respectively. Most of the similarities are still quite high after estimation: The mean cosine values for adjectives are 0.82 for the initial intercept and 0.72 for the minimum-error one. For verbs, the corresponding values are 0.75 and 0.69, respectively. Apparently, the sparsity constraint helps the intercept retaining information from training phrases.

Qualitatively, often the intercept encodes the representation of the original word meaning in

[4]We analyze NMF rather than the better-performing SVD features because the presence of negative values in the latter makes their interpretation very difficult. And NMF achieves comparably good performance for interpretation when dimension exceeds 100.

burst	*significant columns*	*significant rows*
	policeman, mob, guard	hurricane, earthquake, disaster
	Iraqi, Lebanese, Kurdish	conquer, Byzantine, conquest
	jealousy, anger, guilt	policeman, mob, guard
	hurricane, earthquake, disaster	terminus, traffic, interchange
	defender, keeper, striker	convict, sentence, imprisonment
	volcanic, sediment, geological	boost, unveil, campaigner
poisonous	*significant columns*	*significant rows*
	bathroom, wc, shower	ventilation, fluid, bacterium
	ignite, emit, reactor	ignite, emit, reactor
	reptile, mammal, predator	infectious, infect, infected
	ventilation, fluid, bacterium	slay, pharaoh, tribe
	flowering, shrub, perennial	park, lorry, pavement
	sauce, onion, garlic	knife, pierce, brass

Table 6: Interpretability for verbs and adjectives (exemplified by *burst* and *poisonous*).

vector space. For example, if we check the intercept for *poisonous*, the cosine between the original vector space representation (from corpus) and the minimum-error solution intercept (from training phrases) is at 0.7. The NMF dimensions corresponding with the largest intercept entries are rather intuitive for *poisonous*: ⟨*ventilation, fluid, bacterium*⟩, ⟨*racist, racism, outrage*⟩, ⟨*reptile, mammal, predator*⟩, ⟨*flowering, shrub, perennial*⟩, ⟨*sceptical, accusation, credibility*⟩, ⟨*infectious, infect, infected*⟩.

The mathematical reason for the above facts lies in the updating rule of the elastic-net's intercept:

$$\beta_0 = \bar{\mathbf{y}} - \sum_{j=1}^{p} \hat{\beta}_j \bar{\mathbf{x}}_j \qquad (3)$$

Sparsity in the regression coefficients ($\hat{\beta}_j$) encourages intercept β_0 to stay as close to the mean value of response $\bar{\mathbf{y}}$ as possible. So the elastic-net lexical function composition model is *de facto* also capturing the inherent meaning of the functor word, learning it from the training word-phrase pairs. In future research, we would like to test if these lexical meaning representations are as good or even better than standard co-occurrence vectors for single-word similarity tasks.

5 Conclusion

In this paper, we have shown that the lexical function composition model can be improved by advanced regression techniques. We use pathwise coordinate descent optimized elastic-net, testing it on composing intransitive sentences, adjective-noun phrases and determiner phrases in comparison with other composition models, including lexical function estimated with ridge regression. The elastic-net method leads to performance gains on all three tasks. Through sparsity constraints on the model, elastic-net also introduces interpretability in the lexical function composition model. The regression coefficient matrices can often be easily interpreted by looking at large row and column sums, as many matrix entries are shrunk to zero. The intercept of elastic-net regression also plays an interesting role in the model. With the sparsity constraints, the intercept of the model tends to retain the inherent meaning of the word by averaging training phrase vectors.

Our approach naturally generalizes to similar composition tasks, in particular those involving higher-order tensors (Grefenstette et al., 2013), where sparseness might be crucial in producing compact representations of very large objects. Our results also suggest that the performance of the lexical function composition model might be further improved with even more advanced methods, such as nonlinear regression. In the future, we would also like to explore interpretability more in depth, by looking at grouping and interaction effects between features.

Acknowledgments

We acknowledge ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES), and we thank the reviewers for helpful feedback.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Raffaella Bernardi, Georgiana Dinu, Marco Marelli, and Marco Baroni. 2013. A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of ACL (Short Papers)*, pages 53–57, Sofia, Bulgaria.

Stephen Clark. 2013. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd edition*. Blackwell, Malden, MA. In press.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. Least angle regression. *The Annals of statistics*, 32(2):407–499.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with Latent Semantic Analysis. *Discourse Processes*, 25:285–307.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. 2007. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of IWCS*, pages 131–142, Potsdam, Germany.

Tom Griffiths, Mark Steyvers, and Josh Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:211–244.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning, 2nd ed.* Springer, New York.

Darrell Laham. 1997. Latent Semantic Analysis approaches to categorization. In *Proceedings of CogSci*, page 979.

Daniel Lee and Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of NIPS*, pages 556–562.

Scott McDonald and Chris Brew. 2004. A distributional model of semantic context effects in lexical processing. In *Proceedings of ACL*, pages 17–24, Barcelona, Spain.

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

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

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Rob Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *J. Artif. Intell. Res.(JAIR)*, 44:533–585.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.