Collaborative Training of Tensors for Compositional Distributional Semantics

Tamara Polajnar
Computer Laboratory
University of Cambridge
United Kingdom
tamara.polajnar@cl.cam.ac.uk

Abstract

Type-based compositional distributional semantic models present an interesting line of research into functional representations of linguistic meaning. One of the drawbacks of such models, however, is the lack of training data required to train each word-type combination. In this paper we address this by introducing training methods that share parameters between similar words. We show that these methods enable zero-shot learning for words that have no training data at all, as well as enabling construction of high-quality tensors from very few training examples per word.

1 Introduction

Multiple compositional distributional semantic models have been proposed in the past several years. Most models are based on a vector representation for words and a separate function that performs composition on the vectors (Mitchell and Lapata, 2008; Socher et al., 2012; Zanzotto and Dell’arciprete, 2012). Another line of investigation represents atomic types (mainly nouns and sentences) as vectors but predicate types, for example adjectives and verbs, as functions that act upon the atomic types or other functions (Coecke et al., 2011; Baroni et al., 2014; Grefenstette and Sadrzadeh, 2011; Kartsaklis and Sadrzadeh, 2014; Polajnar et al., 2014).

In the full form of the Categorial framework (Coecke et al., 2011), each lexical item and type pairing is represented by a tensor whose order is determined by the number of atomic-type arguments in its category. If we use Combinatory Categorial Grammar (CCG) as the basis for further discussion in this paper, the noun phrase type is $NP$, which is represented by a first order tensor, i.e., a vector. An adjective category is $NP/\cdot/\cdot$ and is modelled by a matrix (a second order tensor), and a transitive verb is a third order tensor whose category is $(S\setminus NP)/NP$. The verb category is interpreted as looking for a noun phrase to the right (object), a noun phrase to the left (subject), and when these are found the function results in a complete sentence $S$. Since the subject and the object are represented by vectors and the verb is a tensor, the natural composition operation is tensor contraction, which is equivalent to matrix multiplication for second order tensors. The result of the composition between the tensor, the subject, and object vectors is a vector representing a composed sentence (Figure 1).

While the type-based approach has strong theoretical grounding (Baroni et al., 2014) and integrates well with Categorial grammars (Coecke et al., 2011; Maillard et al., 2014), there are two major practical challenges. The first is that there are many type-lexeme combinations and some of the types lead to high-order tensors. This results in a large number of parameters that need to be estimated and stored. The second challenge is that some of the type-lexeme combinations are very rare, and, therefore, may never be observed in the training data, leading to difficulties in accurate parameter estimation for those types.

\footnote{In CCG the adjective is correctly represented by $N/N$; however since our datasets do not include determiners, here we make no distinction between $N$ and $NP$.}
In this paper, we propose a method to address the second challenge and we evaluate it both on the ad-
jective and transitive verb case. We build on previous empirical work on the Lexical Function (LF) model (Baroni and Zamparelli, 2010), which considers each adjective to be a matrix that transforms a noun vector into a vector encoding the adjective-noun phrase in the same vector space the original noun, and the contemporaneous Categorial framework (CF) (Coecke et al., 2011), which applies the idea of words as functions to all word classes other than nouns.

2 Related Work

The type-based models have a solid theoretical grounding (Coecke et al., 2011; Baroni et al., 2014) neatly combining mathematical category theory and categorial grammars like CCG with linear algebra to represent the meaning of argument taking words as functions modelled in vector space. The tensors in CF provide a way of representing words like verbs as functions that can recognise the prototypical combinations of features that are expected from their arguments. So, for example, a function representing the verb eat would not only encode that it expects animate subjects that consume edible objects, but ultimately the types of edible objects that make sense with particular subjects. This is different from most other methods (Mitchell and Lapata, 2008; Socher et al., 2012; Zanzotto and Dell’arciprete, 2012) where all words are represented as vectors, and combined with operators of various complexity which themselves do not encode any semantics.

Due to the parameter explosion problems mentioned in Section 1, the CF implementations have mostly been tested on restricted constructions such as adjective-noun or subject-verb-object phrases, while the neural-network-based approaches have been optimised for and tested on full-sentence tasks (Socher et al., 2012; Kiros et al., 2015). By addressing the problem of training data sparsity, we hope to bring the Categorial framework a step closer to full implementation, so that it can be compared to other models on a variety of tasks.

Various approximations of CF which curb the number of parameters have previously been implemented (Grefenstette and Sadrzadeh, 2011; Paperno et al., 2014; Polajnar et al., 2014), but most of those changed the shape of the tensors in some way that diminishes the spirit of the full model. Of these, the Practical Lexical Function (PLF) model of Paperno et al. (2014) comes closest to full sentence implementation of a type-based semantic model. It extends the type-based approach to full sentences by representing argument taking types as a set of matrices, each of which interacts with one argument. While this has the effect of reducing the number of parameters, it also decouples the interactions between arguments, which is one of the main strengths of the tensor-based model. In contrast, Fried et al. (2015) provide a mathematically principled way of achieving parameter reduction while preserving the shape of tensors, and hence the interactions between the arguments, by employing tensor decomposition.

The lack of training data has previously been partially addressed in Grefenstette et al. (2013) by training third-order verb tensors in two steps in order to take advantage of more plentiful verb-object training data. On the other hand, Polajnar et al. (2015) show that it is possible to train full verb tensors with a single-step multi-linear regression method which is the basis of the approaches described in this paper. Bride et al. (2015) introduce the Generalised Lexical Function (GLF), a method for overcoming the sparsity of training data for adjectives. We reimplement this method for comparison in this paper. The two methods that we develop in this paper were loosely inspired by multitask learning (Romera-Paredes et al., 2013) and retrofitting (Faruqui et al., 2015).

3 Methods

In this section we first describe the methods used to train second and third-order tensors as function
representations of adjectives and transitive verbs respectively. We also consider the low-rank representations \cite{fried2015,john2015}, which were demonstrated to produce competitive performance with fewer parameters. We then describe the methods introduced in this paper, that is the full and low-rank training with parameter sharing. And finally, we describe the GLF model \cite{fried2015}, which is introduced as a competitive baseline for the adjective methods. All methods have been implemented using gradient descent, rather than analytic regression.

3.1 Basic Tensor Training

3.1.1 Adjective

We model each adjective as a linear function that maps a noun to an adjective-noun phrase. The \( N \)-dimensional noun vectors \( (n) \) are transformed into \( N \)-dimensional noun phrase vectors via an \( N \times N \) matrix \( A \), just as in the Lexical Function (LF) model of Baroni and Zamparelli \cite{baroni2010}. The loss function consists of minimising the error between the vector resulting from the adjective-noun multiplication and a distributional vector representing the adjective-noun phrase \( (z_{an}) \):

\[
L(A) = \|A n - z_{an}\|_2
\]

Low-rank adjectives We learn low-rank adjective matrices by fixing a maximal rank \( R \) and maintaining each matrix in a rank-decomposed form, which is similar to the singular value decomposition (SVD) \cite{fried2015}. The low-rank representation for an adjective \( A \) is

\[
A = \sum_{r=1}^{R} U_r \otimes V_r
\]

where \( U \in \mathbb{R}^{R \times N}, V \in \mathbb{R}^{R \times N} \) are parameter matrices, \( U_r \) gives the \( r \)th row of matrix \( U \), and \( \otimes \) is the tensor product.

The adjective matrix’s action on vectors is then given by

\[
A n = U^T (V n)
\]

3.1.2 Verb

We model each transitive verb as a bilinear function mapping subject and object noun vectors, each of dimensionality \( N \), to a single sentence vector of dimensionality \( S \). Each transitive verb \( V \) is associated with a third-order tensor \( V \in \mathbb{R}^{S \times N \times N} \), which defines this bilinear function. If vectors \( n_s \in \mathbb{R}^N, n_o \in \mathbb{R}^N \) for subject and object nouns, respectively, then the loss function for each verb is:

\[
L(V) = \|n_s V n_o - z_s\|_2
\]

That is the error between the sentence vector produced by applying tensor contraction (the higher-order analogue of matrix multiplication) to the verb tensor and two noun vectors and the distributional representation for that sentence \( z_s \).

Polajnar et al. \cite{polajnar2015} examine several different distributional sentence spaces; from these we chose the intra-sentential contextual sentence space consisting of content words that occur within the same sentences as the SVO triple, disregarding the verb itself.

Low-rank verbs Following Fried et al. \cite{fried2015}, we use canonical polyadic (CP) decomposition representation of verb tensors. CP decomposition factors a tensor into a sum of \( R \) tensor products of vectors, reducing the number of parameters we have to learn.

The low-rank representation for a verb \( V \) is:

\[
V = \sum_{r=1}^{R} P_r \otimes Q_r \otimes R_r
\]

where \( P \in \mathbb{R}^{R \times S}, Q \in \mathbb{R}^{R \times N}, R \in \mathbb{R}^{R \times N} \) are parameter matrices.

Representing tensors in this form allows us to avoid explicitly generating the full tensor by formulating the verb tensor’s action on noun vectors as matrix multiplication:

\[
V(s, o) = P^T (Q s \odot R o)
\]

where \( \odot \) is the elementwise vector product. As a result we are able to reduce the number of parameters needed to model each verb from \( S \times N \times N \) to \( R \times (2N + S) \), which in our experiments where we have \( R = 50 \) and \( N = S = 100 \) means a reduction from 1,000,000 to 15,000 parameters per verb.

3.2 Collaborative Tensor Training

While low-rank methods reduce the amount of memory required to store a full lexicon and the
amount of time required to train the tensors, they do not address the problem of data sparsity. To train tensors we need high-quality examples, which potentially have to be extracted from parsed data. However, there are word-types for which there are few reliable training examples and others for which there are no training examples at all. In those cases we would still like to have a non-zero approximation for the particular function.

We propose two approaches to address these instances of sparsity. Both approaches are based on the existence of an external method that gives similarity between the words for which we are trying to build tensors. We define this as a function $\phi(w_1, w_2)$ which gives us the similarity between words $w$ corresponding to tensors $T_1$ and $T_2$, where $T$ refers to either adjective matrices or verb third-order tensors as the approaches are analogous across the types. In addition, we only use the $K$ top most similar tensors according to $\phi$, where $K$ is currently a manually chosen parameter. The method which provides the similarities in $\phi$ could be manual or derived from an ontology or any distributional or distributed representation of these words. If the method relies on vectors, then these do not have to match the training data at all, as we only rely on a matrix of similarity values between all pairs of adjectives (or separately verbs) that we are training.

The first approach shares parameters between tensors that we have declared to be similar by directly creating a weighted average of the target tensor with the sum of the tensors of the $K$ closest words (weighted by the similarity values from $\phi$). This approach is somewhat heavy handed and we expect that it produces large jumps around the error landscape which is being navigated by gradient descent. The second approach is more gentle and uses a regularisation component to push a tensor closest to its nearest neighbours (according to $\phi$) by encouraging smaller distances between them. In the experimental sections we apply these two methods individually and together.

We compare our methods to the GLF model \cite{Bride2015} on the adjective-based datasets. GLF is a third order tensor that is used to generate adjective matrices using adjective vectors. It is trained using Lexical Function adjective matrices. The advantage of our methods over GLF is that we are not restricted by the source of our word-word similarities, where GLF requires adjective vectors that are sourced from the same corpus as the lexical function training data. Word vectors created from another corpus, even with the same method, may lie in a different vector space and thus be incompatible with the tensor and matrices trained in the original space.

In addition we introduce the deterministic function $\phi$ and two tuneable scalar parameters each of
Figure 3: The variation in parameters $\alpha$ and $\beta$ which regulate PS and FT, respectively, as the number of adjective-noun training pairs per adjective is increased (AB2). The Spearman results shown are calculated using full matrices (top row) and low-rank adjectives (bottom row) on ML10 data.

which balances the contributions of one of the methods. On the other hand, GLF requires construction of an additional $N \times N \times N$ sized tensor, which in the verb case would have to be extended to a fourth order tensor.

3.2.1 Parameter Sharing

In this first method we share parameters between most similar tensors using the function $\phi$. We adjust the appropriate loss function (Eq. 1 or Eq. 4) to incorporate parameter sharing (PS) during gradient descent:

$$L_{ps}(T) = (1 - \alpha)L(T) + \frac{\alpha}{K} \sum_{i=1}^{K} \phi(w, w_i)T_i \quad (7)$$

The parameter $\alpha$ balances the amount of tensor we are replacing by the normalised sum of the nearby tensors. In case of the low-rank representations of tensors, we use the deconstructed versions of the tensors and share the parameters between corresponding decomposed matrix representations by aligning the $U$, $V$, and for verbs $W$, for the word pairs without reconstructing the tensors.

3.2.2 Fitting

The second method is used in place of l2-regularisation to push the parameters of the current tensor closer to the parameters of the tensors of the similar words. The regularisation component is

$$R_{ft}(T) = \frac{\beta}{K} \sum_{i=1}^{K} \phi(w, w_i)||T - T_i||_F \quad (8)$$

and is integrated into the training function via the parameter $\beta$:

$$L_{ft}(T) = L(T) + R_{ft}(T) \quad (9)$$

Like with PS, in the low-rank representations we regularise each of the component matrices separately.

3.2.3 Generalised Lexical Function

The GLF model (Bride et al. 2015) addresses training data sparsity by introducing a third order tensor ($A$) that acts as a function that takes in an adjective vector $a_i$ and generates a matrix $A_i$ for that adjective. We train the tensor using gradient descent to minimise the error between the adjective matrix produced by GLF and the adjective matrices generated by the Lexical Function ($A^{LF}$), as described in Equation 11

$$L(A) = \sum_i ||Aa_i - A_i^{LF}||_F \quad (10)$$

We vary slightly from the loss described by Bride et al. (2015) in that we use the standard Frobenius norm instead of the straight subtraction.

The difference between the two methods we introduced and GLF is that GLF does require adjective
vectors which are created by the same procedure as the adjective training data; therefore, if an adjective is completely unavailable in the distributional training corpus, this method could not produce a matrix representation for that word.

For the sparse experiments we train the GLF as above, but $A^{LF}$ are reconstituted low-rank matrices. Therefore, this method will still have many more parameters than the true low-rank approximations.

4 Experimental Settings

4.1 Test Datasets

We use several datasets that test composition or directly compare the quality of the produced tensors:

- **ML10**: adjective-noun (AN) pairs rated for similarity (Mitchell and Lapata, 2010).
- **MEN**: word-word pairs rated for relatedness from which we extract the adjective-adjective pairs only (Bruni et al., 2014).
- **SIMLEX**: word-word pairs rated for similarity from which we extracted the adjective-adjective and verb-verb pairs (Hill et al., 2014).
- **GS11**: a verb disambiguation dataset consisting of subject-verb-object (SVO) triples arranged in pairs, where in each pair the subject and the object remain the same but the verb changes (Greffestette and Sadrzadeh, 2011).
- **KS14**: a dataset subject-verb-object sentence pairs rated for similarity (Kartsaklis and Sadrzadeh, 2014), which is an extension of the verb-object component of the ML10 dataset.
- **ANVAN**: a verb disambiguation dataset containing pairs of adjective-noun-verb-adjective-noun sentences where only the verb varies (Kartsaklis et al., 2013).

4.2 Training Data

In order to train the tensors for the adjectives and verbs occurring in the above test data we need to find examples of their usage in text. We use the October 2013 dump of Wikipedia articles, which was tokenised using the Stanford NLP tools[^2]lemmatised with the Morpha lemmatiser (Minnen et al., 2001), and parsed with the C&C parser (Clark and Curran, 2007).

We use the parser output to find adjective-noun and subject-verb-object combinations that involve our target words. From these we choose up to 500 tuples that contain nouns that occur at least 100 times and which themselves occur at least twice. Some words are quite rare and do not have any training data, e.g. the adjective ashamed, or very little training data, e.g. adjectives glad and gritted, each of which has a single training example.

The vectors for nouns and the holistic vectors for the AN and SVO phrases are generated using the Paragraph Vector (Le and Mikolov, 2014) model from the modified word2vec (Mikolov et al., 2013) code[^3].

4.3 Model Training

The LF model forms the basis of all the collaborative training models. The adjectives and verbs are trained up to 200 iterations using batched gradient descent with ADADELTA (Zeiler, 2012), at which point most of the tensors have finished training. The stopping criterion for adjectives is stagnation or an increase in training error. For verbs we also use a 10% validation dataset if there are at least 20 training points. Full tensor training also uses $l_2$ regularisation with parameter 0.1, while sparse tensors are trained without regularisation as this was observed to be more optimal in Fried (2015).

We then use the pre-trained adjective matrices to train the GLF model for 10,000 iterations. The original GLF algorithm specifies training until convergence, but we found little improvement and sometimes observed overfitting with further training. The combined PS and FT models are trained for up to 200 iterations using the same stopping criteria as above. More training did not lead to significant improvements.

5 Adjective Ablation Experiments

We use ML10 as a development dataset and test a range of mixing parameters for PS and FT, and how they work together. There can be two types of sparsity that occur in data: inadequate amount of training data (AB1) and complete lack of training data (AB2). We implement both of these test envi-
environments separately in order to observe the effects on the algorithms. In both cases, we assume that: we have some information about adjective-adjective similarities; the non-zero adjectives are pre-trained for 200 epochs and then for 200 more epochs using a combined PS and FT loss function; and the parameters which are being shared are updated after each epoch.

5.1 Tuning Experiments

In these experiments we cycle through a range of values for PS and FT parameters $\alpha$ and $\beta$, while testing between 1 and 70% of adjectives with no training data (AB1) and between 1 and 70% of training data per adjective (AB2).

Adjectives can lack all training data if we have no adjective-noun pairs that occur with sufficient frequency to be automatically chosen as good-quality candidates for training. We vary the percentage of adjectives for which we keep all the available training pairs. Figure 2 shows the results of Spearman correlation on ML10 data when at most 1, 5, and 70% of 297 adjectives are retained. Each line in the graph represents a particular PS parameter $\alpha \in \{0, 0.1, 0.5, 0.9, 1\}$. The x-axis is regulated by variation of the FT parameter $\beta \in \{0, 0.01, 0.05, 0.1\}$. The y-axis shows the Spearman correlation on the ML10 dataset of adjectives trained with a particular combination of $\alpha$ and $\beta$.

Along the y-axis ($x = \beta = 0$) we can observe the effects of the PS without FT, while the blue dashed line represents the effects of FT without PS ($\alpha = 0$). The point where the blue line crosses the y-axis ($\alpha = \beta = 0$) is equivalent to the performance of the Lexical Function.

Variation in FT parameter ($\beta$) shows little improvement if PS is being used ($\alpha > 0$) on full tensor tuning results (top row Figure 2). The more training data is available the better higher values of $\alpha$ do; although disregarding the adjective’s training data by only using the mix of nearest neighbour adjective matrices ($\alpha = 1$, magenta dashed line) is counterproductive. This indicates that even when plenty of training data is available, the knowledge of similarities between words contributes towards training.

Low-rank tensors are in general more sensitive to both types of collaborative tensor training (bottom row Figure 2). Low-rank matrices have a fraction of the parameters of full tensors and, along with no training data for a sizeable number of adjectives this leads to a very sparse model. Note also that both FT and PS also rely on an assumption that the vectors that constitute ranks of a particular adjective correspond the same rank in another adjective. This assumption is not necessarily correct, but the fact that the methods lead to some success may indicate that they force the vectors to align through training. When there are very few adjectives with training data (AB1), the best performing settings for full tensors appear to be $\alpha \in 0, 0.1$ and with more training data available the PS parameter increases to 0.9, while $\beta = 0.01$ performs consistently well. More stringent tuning could be accomplished by randomised repeats of experiments, but these are too costly.

The second experiment demonstrates what happens when there are few training examples per adjective. Figure 3 shows the results of Spearman correlation on ML10 data when at most 1, 5, and 70% of the maximum of 500 training pairs per adjective are retained. Here we see that once we have any training data at all the settings that worked in the previous experiment with high amounts of training data also apply here and $\alpha = 0.9$ consistently performs well.

In the low-rank AB2 experiments (bottom row Figure 3) Spearman values vary widely as both PS and FT parameters change, but overall there appears to be a convergence of performance as the amount of training data is increased. Variations in the lines along the x-axis show that low-rank tensors are more sensitive to FT than full tensors, which is consistent with the finding that low-rank tensors can perform worse with regularisation (Fried, 2015). Unlike with the full-rank results, the visual inspection of the low-rank graphs in Figures 2 and 3 does not lead to obvious best settings. For that reason we examine ($\alpha$, $\beta$) pair value rankings across all the settings in both experiments and find that low $\alpha$ values paired with higher $\beta$s work well. The pairing that ranks most highly is (0, 0.1), followed by (0.1, 0.1).

5.2 Testing

In the previous section we observed the behaviour of parameter sharing and fitting as training data sparsity and parameter values varied. These tuning experiments allow us to pick a set of parameter val-
null.

We test for adjective-noun composition (ML10), word-word relatedness (MEN) and similarity (SIMLEX). We compare the composed AN vectors and the unfurled adjective matrices using cosine similarity (MEN) and similarity (SIMLEX). We compare the composed AN vectors and the unfurled adjective matrices using cosine similarity (MEN) and similarity (SIMLEX).

Tables 1 and 2 report the testing results for AB1. Grey cells indicate values that were used as test data during parameter tuning, while grey numbers indicate non-significant correlations. Highest values in each section and column are bolded. ML10 additive baseline is 0.52, and the vector similarity baseline is 0.48 for MEN and 0.37 for SIMLEX on 100% of the data.

For low-rank tensors we find that variation is not clearly definable and, therefore, chose two fixed settings corresponding to the two highest ranking parameter pairs (0, 0.1), PS+FT\textsubscript{fix}, and (0.1, 0.1), PS+FT\textsubscript{var}. The former setting is just FT as parameter sharing is null.

We test for adjective-noun composition (ML10), word-word relatedness (MEN) and similarity (SIMLEX). We compare the composed AN vectors and the unfurled adjective matrices using cosine similarity.

Tables 1 and 2 report the testing results for AB1 and AB2 with the above settings and compare them with GLF and LF. In the grey cells we have the values that appeared in Figures 2 and 3 which were considered when choosing the parameter settings. The rest of the values in the tables represent testing results without tuning on these datasets. The light grey numbers are not significantly correlated with the data with $p < 0.05$. Some negative correlations occur in the testing results, and we will examine these more closely in Section 7.

In Table 1, the columns that are marked with 70% or less on all datasets show that LF requires at least some training data for each adjective. GLF performs well when there is moderate sparsity in the percentage of available adjectives (5%-70%). It also performs better than PS+FT with low-rank adjective matrices on ML data, but it also has many more parameters. PS+FT gets highest results on MEN and SIMLEX more often than GLF, and most significantly both GLF and PS+FT regularly outperform LF even when all the training data is available.

The additive baseline on ML10 with the word2vec vectors is difficult to beat, although PS+FT does match it occasionally. More interestingly the collaborative representations do better on word-word relatedness (MEN) and similarity (SIMLEX) than both
Table 3: Spearman values as the percentage of verbs with training data is increased (AB1). Grey numbers indicate non-significant correlations.

| Method | Full Tensor | KS14 | GS11 | ANVAN | SIMLEX |
|--------|-------------|------|------|-------|--------|
|        | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% |
| PS+FT $\text{fix}_3$ | 0.35, 0.29, 0.34, 0.39, 0.50 | 0.03, 0.08, 0.35, 0.25, 0.31 | 0.11, 0.04, 0.08, 0.07, 0.18 | 0.16, 0.33, 0.27, 0.12, 0.12 |
| PS+FT $\text{var}_3$ | 0.44, 0.37, 0.33, 0.39, 0.50 | 0.02, 0.05, 0.35, 0.25, 0.31 | 0.14, 0.06, 0.10, 0.07, 0.18 | 0.16, 0.36, 0.29, 0.12, 0.12 |
| Tensor | - - 0.11 0.33 0.54 | - - -0.06 0.29 0.38 | - - 0.07 0.10 0.14 | - - 0.03 0.02 0.02 |

Table 4: Spearman values as the percentage of training data per verb is increased (AB2). Highest values in each column are bolded, while grey numbers indicate non-significant correlations.

| Method | Full Tensor | KS14 | GS11 | ANVAN | SIMLEX |
|--------|-------------|------|------|-------|--------|
|        | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% | 1% 5% 30% 70% 100% |
| PS+FT $\text{fix}_2$ | 0.19, 0.19, 0.05, 0.45, 0.54 | 0.25, 0.26, 0.17, 0.24, 0.44 | -0.01, -0.01, -0.07, 0.13, 0.12 | -0.06, 0.19, 0.18, 0.17, 0.09 |
| PS+FT $\text{fix}_3$ | -0.03, -0.11, 0.17, 0.26, 0.38 | 0.02, 0.02, 0.24, 0.16, 0.23 | -0.05, 0.02, 0.09, 0.11, 0.10 | 0.40, 0.33, 0.13, 0.08, 0.01 |
| Tensor | - - -0.04 0.25 0.52 | - - -0.08 0.23 0.35 | - - 0.07 0.15 | - - 0.01 0.07 0.08 |

6 Verb Experiments

We repeat ablation testing with the 285 verbs that occur in the verb datasets that cover: SVO relatedness (KS14), verb disambiguation within a noun-verb-noun (GS11) and an adjective-noun-verb-adjective-noun SVO contexts (ANVAN), and verb-verb similarity (SIMLEX). Table 3 shows results as the number of verbs with training data is increased (AB1) and Table 4 shows results as the number of training examples per verb is increased (AB2). The additive baseline results on 100% of the data are 0.59 for KS14, 0.13 for GS11, 0.03 for ANVAN, and the vector similarity baseline is 0.14 for SIMLEX. We see large improvements over the low GS11, ANVAN, and SIMLEX baselines; however, the additive baseline on KS14 which is an extension of the ML10 verb-object dataset is difficult to beat. This is constant with previous findings, as are the scores we achieve here (Fried et al., 2015).

The disambiguation datasets are interesting because they seem to be a harder test for composition than the relatedness dataset (KS14). In particular the ANVAN dataset permits us to combine the collaboratively trained adjectives and verbs into a single test. In the original paper, Kartsaklis et al. (2013) provided a similarly low baseline with addition and distributional vectors on the ANVAN dataset, but found an improvement in performance after using sense-disambiguated verbs. They achieved a maximum Spearman ρ of 0.28 when considering the disambiguated verb vectors only and 0.23 when using addition to produce the composed ANVAN phrases. We get 0.20 as the highest values using adjective matrices and verb tensors for composition without any prior disambiguation. As both GS11 and ANVAN are testing for disambiguation, it may indicate that tensors themselves encode a range of senses. This conjecture is also supported by findings that there is a mild correlation between the tensor rank, in low-rank representations, and the number of WordNet senses for a word (Fried, 2015).

We use the PS+FT settings tuned for matrices on ML10 data, and as a result we see greater variability in performance, with the original tensor model often outperforming the collaborative training meth-
Table 5: Nearest neighbours of two adjectives and two verbs with and without collaborative training.

| yellow     | play     |
|------------|----------|
| PS+FT LF   | PS+FT Tensor |
| orange     | red      | participate make start do |
| red        | blue     | red blue |
| blue       | white    | blue white |
| coloured   | green    | do join |
| brown      | brown    | win get |

| outdoor    | entangle |
|------------|----------|
| PS+FT LF   | PS+FT Tensor |
| domestic   | win-over transmit |
| local      | large    | win-over transit |
| outer      | various  | disorganize deposit |
| new        | small    | multiply eat |
| foreign    | new      | tap wash |

7 Qualitative Analysis

Since the goal of training tensors for the Categorial framework is good performance in composition tasks, we tuned our parameters on an adjective-noun composition dataset (ML10). In Table 1 we can see that when we have only 2 adjectives with training data (1% column) we have acceptable performance on this dataset; but, the performance on the datasets where we compare the adjective matrices directly (MEN and SIMLEX) is statistically non-correlated and sometimes negative. Nearest neighbours analysis of adjectives produced by the parameter settings $fix_1$ and $var$ shows that similarities between all adjectives approach one and lead to nonsensical rankings. This is due to the fact that the numbers in the matrices themselves are very low and approach machine precision. Using cosine similarity leads to elementwise multiplication between low numbers and hence near-zero values in the numerator. On the other hand, composition is performed using matrix multiplication between the adjective matrices and the noun vectors, which have much larger numbers, resulting in the more sensible performance in the composed datasets. An alternative way of evaluating the quality of the tensors would be to treat them as functions, and instead of cosine employ the function comparison within the type-driven framework introduced by Maillard and Clark (2015).

Another interesting phenomenon we observed is that PS+FT works better than LF when all adjective training data is available. So if we compare the nearest neighbours for adjectives produced by $fix_1$ setting to the ones produced with LF, we can notice qualitative improvement (Table 5). Keeping in mind that our pool of nearest neighbours is limited to the 297 adjectives in our training data, we can see subtle differences in the adjective yellow, which is well represented with a large amount of training data and nearest neighbours. Although all of the closest terms are chromatic, orange and red are the closest when collaborative training is involved. For the underrepresented adjective outdoor we can see that PS+FT finds more semantically related and less general adjectives although true neighbours are not available.

For verbs we found that PS+FT often does worse than the straight-forward tensor method (Tables 3 and 4). In Table 5 we can see an example of easy to train verb to play where PS+FT does indeed rate a similar term participate highly. In contrast, the verb to entangle is rarer, and hence would have less training data and poorer vector representation. The closest term is win-over, a verb which is artificially hyphenated in the ANVAN dataset and hence has no naturally occurring training data. Together adjective and verb results indicate that a larger training pool from which we can choose related tensors may produce a better representations overall.

8 Discussion

In this paper we introduced two methods that address the lack of training data in the type-driven framework for compositional distributional semantics. In our experiments we use distributed vectors which have been found to achieve state-of-the-art results on some of the datasets we used here (Fried et al., 2015); however, the goal here was to compare these methods to the standard regression approach where each tensor is trained separately. We find that for both full tensors and low-rank approximations collaborative training enables training of tensors under conditions where individual training would result in low-quality or null tensors. In addition these methods often outperform the individual training even with all of the available training data.
References

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 (EMNLP).

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. Linguistic Issues in Language Technology, 9.

Antoine Bride, Tim Van de Cruys, and Nicholas Asher. July 2015. A generalisation of lexical functions for composition in distributional semantics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 281–291, Beijing, China. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1028

Elia Bruni, Nam Khanh Tran, and Marco Baroni. January 2014. Multimodal distributional semantics. J. Artif. Int. Res., 49(1):1–47. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=2655713.2655714

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics, 33(4): 493–552.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2011. Mathematical foundations for a compositional distributional model of meaning. Linguistic Analysis, 36(1-4):345–384.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In HLT-NAACL, pages 1606–1615.

Daniel Fried. Low-rank tensor approximations for compositional distributional semantics. Master’s thesis, University of Cambridge, 2015.

Daniel Fried, Tamara Polajnar, and Stephen Clark. July 2015. Low-rank tensors for verbs in compositional distributional semantics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 731–736, Beijing, China. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2120

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimenting with transitive verbs in a DisCo-Cat. Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics. URL http://www.aclweb.org/anthology-new/W/W11/W11-2507.pdf

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013).

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. Computational Linguistics, abs/1408.3456. URL http://arxiv.org/abs/1408.3456

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. June 2014. A study of entanglement in a categorical framework of natural language. In Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL), Kyoto, Japan.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. August 2013. Separating disambiguation from composition in distributional semantics. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pages 114–123, Sofia, Bulgaria. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3513

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. CoRR, abs/1506.06726. URL http://arxiv.org/abs/1506.06726

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In International Conference on Machine Learning (ICML).

Jean Maillard and Stephen Clark. July 2015. Learning adjective meanings with a tensor-based skip-gram model. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, pages 327–331, Beijing, China. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K15-1035

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A type-driven tensor-based semantics for CCG. EACL 2014 Type Theory and Natural Language Semantics Workshop. URL http://goo.gl/JTEKLR

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Neural Information Processing Systems (NIPS).
Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34:1388–1439.

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

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 90–99.

Tamara Polajnar, Luana Fagarasan, and Stephen Clark. 2014. Reducing dimensions of tensors in type-driven distributional semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1036–1046.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2015. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, Lisbon, Portugal.

Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. May 2013. Multilinear multitask learning. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1444–1452. JMLR Workshop and Conference Proceedings. URL [http://jmlr.org/proceedings/papers/v28/romera.pdf](http://jmlr.org/proceedings/papers/v28/romera.pdf).

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fabio M. Zanzotto and Lorenzo Dell’arciprete. 2012. Distributed tree kernels. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 193–200.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*. 
