Improving Distributional Semantic Vectors through Context Selection and Normalisation

Tamara Polajnar
University of Cambridge
Computer Laboratory
tp366@cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
sc609@cam.ac.uk

Abstract

Distributional semantic models (DSMs) have been effective at representing semantics at the word level, and research has recently moved on to building distributional representations for larger segments of text. In this paper, we introduce novel ways of applying context selection and normalisation to vary model sparsity and the range of values of the DSM vectors. We show how these methods enhance the quality of the vectors and thus result in improved low dimensional and composed representations. We demonstrate these effects on standard word and phrase datasets, and on a new definition retrieval task and dataset.

1 Introduction

Distributional semantic models (DSMs) (Turney and Pantel, 2010; Clarke, 2012) encode word meaning by counting co-occurrences with other words within a context window and recording these counts in a vector. Various IR and NLP tasks, such as word sense disambiguation, query expansion, and paraphrasing, take advantage of DSMs at a word level. More recently, researchers have been exploring methods that combine word vectors to represent phrases (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010) and sentences (Coecke et al., 2010; Socher et al., 2012). In this paper, we introduce two techniques that improve the quality of word vectors and can be easily tuned to adapt the vectors to particular lexical and compositional tasks.

The quality of the word vectors is generally assessed on standard datasets that consist of a list of word pairs and a corresponding list of gold standard scores. These scores are gathered through an annotation task and reflect the similarity between the words as perceived by human judges (Bruni et al., 2012). Evaluation is conducted by comparing the word similarity predicted by the model with the gold standard using a correlation test such as Spearman's ρ.

While words, and perhaps some frequent shorter phrases, can be represented by distributional vectors learned through co-occurrence statistics, infrequent phrases and novel constructions are impossible to represent in that way. The goal of compositional DSMs is to find methods of combining word vectors, or perhaps higher-order tensors, into a single vector that represents the meaning of the whole segment of text. Elementary approaches to composition employ simple operations, such as addition and elementwise product, directly on the word vectors. These have been shown to be effective for phrase similarity evaluation (Mitchell and Lapata, 2010) and detection of anomalous phrases (Kochmar and Briscoe, 2013).

The methods that will be introduced in this paper can be applied to co-occurrence vectors to produce improvements on word similarity and compositional tasks with simple operators. We chose to examine the use of sum, elementwise product, and circular convolution (Jones and Mewhort, 2007), because they are often used due to their simplicity, or as components of more complex models (Zanzotto and Dell'Arciprete, 2011).

The first method is context selection (CS), in which the top N highest weighted context words per vector are selected, and the rest of the values are discarded (by setting to zero). This technique is similar to the way that Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) selects the number of topics that represent a word, and the word filtering approach in Gamallo and Bordag (2011). It has the advantage of improving word representations and vector sum representations (for compositional tasks) while using vectors with fewer non-zero elements. Programming languages often have efficient strategies for stor-

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 230–238,
Gothenburg, Sweden, April 26-30 2014.

ing these *sparse* vectors, leading to lower memory usage. As an example of the resulting accuracy improvements, when vectors with up to 10,000 non-zero elements are reduced to a maximum of $N = 240$ non-zero elements, the Spearman ρ improves from 0.61 to 0.76 on a standard word similarity task. We also see an improvement when used in conjunction with further, standard dimensionality reduction techniques: the CS sparse vectors lead to reduced-dimensional representations that produce higher correlations with human similarity judgements than the original full vectors.

The second method is a weighted l^2-normalisation of the vectors prior to application of singular value decomposition (SVD) (Deerwester et al., 1990) or compositional vector operators. It has the effect of drastically improving SVD with 100 or fewer dimensions. For example, we find that applying normalisation before SVD improves correlation from $\rho = 0.48$ to $\rho = 0.70$ for 20 dimensions, on the word similarity task. This is an essential finding as many more complex models of compositional semantics (Coecke et al., 2010; Baroni and Zamparelli, 2010; Andreas and Ghahramani, 2013) work with tensor objects and require good quality low-dimensional representations of words in order to lower computational costs. This technique also improves the performance of vector addition on texts of any length and vector elementwise product on shorter texts, on both the similarity and definitions tasks.

The definition task and dataset are an additional contribution. We produced a new dataset of words and their definitions, which is separated into nine parts, each consisting of definitions of a particular length. This allows us to examine how compositional operators interact with CS and normalisation as the number of vector operations increases.

This paper is divided into three main sections. Section 2 describes the construction of the word vectors that underlie all of our experiments and the two methods for adaptation of the vectors to specific tasks. In Section 3 we assess the effects of CS and normalisation on standard word similarity datasets. In Section 4 we present the compositional experiments on phrase data and our new definitions dataset.

2 Word Vector Construction

The distributional hypothesis assumes that words that occur within similar contexts share similar meanings; hence semantic vector construction first requires a defintition of context. Here we use a window method, where the context is defined as a particular sequence of words either side of the target word. The vectors are then populated through traversal of a large corpus, by recording the number of times each of the target words co-occurs with a context word within the window, which gives the raw target-context co-occurrence frequency vectors (Freq).

The rest of this section contains a description of the particular settings used to construct the raw word vectors and the weighting schemes (tTest, PPMI) that we considered in our experiments. This is followed by a detailed description of the context selection (CS) and normalisation techniques. Finally, dimensionality reduction (SVD) is proposed as a way of combating sparsity and random indexing (RI) as an essential step of encoding vectors for use with the convolution operator.

Raw Vectors We used a cleaned-up corpus of 1.7 billion lemmatised tokens (Minnen et al., 2001) from the October, 2013 snapshot of Wikipedia, and constructed context vectors by using sentence boundaries to provide the window. The set of context words C consisted of the 10,000 most frequent words occurring in this dataset, with the exception of stopwords from a standard stopword list. Therefore, a frequency vector for a target word $w_i \in W$ is represented as $\vec{w_i} = \{f_{w_i c_j}\}_j$, where $c_j \in C$ ($|C| = 10,000$), W is a set of target words in a particular evaluation dataset, and $f_{w_i c_j}$ is the co-occurrence frequency between the target word, w_i and context word, c_j.

Vector Weighting We used the tTest and PPMI weighting schemes, since they both performed well on the development data. The vectors resulting from the application of the weighting schemes are as follows, where the tTest and PPMI functions give weighted values for the basis vector corresponding to context word c_j for target word w_i:

$$\text{tTest}(\vec{w_i}, c_j) = \frac{p(w_i, c_j) - p(w_i)p(c_j)}{\sqrt{p(w_i)p(c_j)}} \quad (1)$$

$$\text{PPMI}(\vec{w_i}, c_j) = p(w_i, c_j) \log \left(\frac{p(w_i, c_j)}{p(w_i)p(c_j)} \right) \quad (2)$$

where $p(w_i) = \frac{\sum_j f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$, $p(c_j) = \frac{\sum_i f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$, and $p(w_i, c_j) = \frac{f_{w_i c_j}}{\sum_k \sum_l f_{w_k c_l}}$.

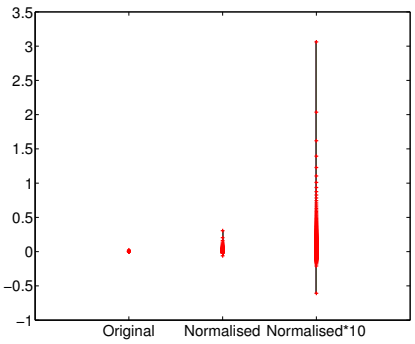

Figure 1: The range of context weights on tTest weighted vectors before and after normalisation.

Context Ranking and Selection The weighting schemes change the importance of individual target-context raw co-occurrence counts by considering the frequency with which each context word occurs with other target words. This is similar to term-weighting in IR and many retrieval functions are also used as weighting functions in DSMs. In the retrieval-based model ESA (Gabrilovich and Markovitch, 2007), only the N highest-weighted contexts are kept as a representative set of "topics" for a particular target word, and the rest are set to zero. Here we use a similar technique and, for each target word, retain only the N-highest weighted context words, using a word-similarity development set to choose the N that maximises correlation across all words in that dataset. Throughout the paper, we will refer to this technique as context selection (**CS**) and use N to indicate the maximum number of contexts per word. Hence all word vectors have at most N non-zero elements, effectively adjusting the sparsity of the vectors, which may have an effect on the sum and elementwise product operations when composing vectors.

Normalisation PPMI has only positive values that span the range $[0, \infty]$, while tTest spans $[-1, 1]$, but generally produces values tightly concentrated around zero. We found that these ranges can produce poor performance due to numerical problems, so we corrected this through weighted row normalisation: $\vec{w} := \lambda \frac{\vec{w}}{||\vec{w}||_2}$. With $\lambda = 10$ this has the effect of restricting the values to $[-10, 10]$ for tTest and $[0, 10]$ for PPMI. Figure 1 shows the range of values for tTest. In general we use $\lambda = 1$, but for some experiments we use $\lambda = 10$ to push the highest weights above 1, as a way of combating the numerical errors that are likely to arise due to repeated multiplications of small numbers. This normalisation has no effect on the ordering of context weights or cosine similarity calculations between single-word vectors. We apply normalisation prior to dimensionality reduction and RI.

SVD SVD transforms vectors from their target-context representation into a target-topic space. The resulting space is *dense*, in that the vectors no longer contain any zero elements. If $\mathbf{M}$ is a $|w| \times |C|$ matrix whose rows are made of word vectors $\vec{w_i}$, then the lower dimensional representation of those vectors is encoded in the $|W| \times K$ matrix $\hat{\mathbf{M}}_K = \mathbf{U}_K \mathbf{S}_K$ where $\text{SVD}(\mathbf{M}, K) = \mathbf{U}_K \mathbf{S}_K \mathbf{V}_K$ (Deerwester et al., 1990). We also tried non-negative matrix factorisation (NNMF) (Seung and Lee, 2001), but found that it did not perform as well as SVD. We used the standard Matlab implementation of SVD.

Random Indexing There are two ways of creating RI-based DSMs, the most popular being to initialise all target word vectors to zero and to generate a random vector for each context word. Then, while traversing through the corpus, each time a target word and a context word co-occur, the context word vector is added to the vector representing the target word. This method allows the RI vectors to be created in one step through a single traversal of the corpus. The other method, following Jones and Mewhort (2007), is to create the RI vectors through matrix multiplication rather than sequentially. We employ this method and assign each context word a random vector $\vec{e_{c_j}} = \{r_k\}_k$ where r_k are drawn from the normal distribution $\mathcal{N}(0, \frac{1}{D})$ and $|\vec{e_{c_j}}| = D = 4096$. The RI representation of a target word $RI(\vec{w_i}) = \vec{w_i}\mathbf{R}$ is constructed by multiplying the word vector $\vec{w_i}$, obtained as before, by the $|C| \times D$ matrix $\mathbf{R}$ where each column represents the vectors $\vec{e_{c_j}}$. Weighting is performed prior to random indexing.

3 Word Similarity Experiments

In this section we investigate the effects of context selection and normalisation on the quality of word vectors using standard word similarity datasets. The datasets consist of word pairs and a gold standard score that indicates the human judgement of the similarity between the words within each pair. We calculated the similarity between word vectors for each pair and compared our results with the gold standard using Spearman correlation.

	tTest		PPMI		Freq	
Data	Max ρ	Full ρ	Max ρ	Full ρ	Max ρ	Full ρ
MENdev†	**0.75**	0.73	**0.76**	0.61	**0.66**	0.57
MENtest	**0.76**	0.73	**0.76**	0.61	**0.66**	0.56
WS353	**0.70**	0.63	**0.70**	0.41	**0.57**	0.41

Table 1: Values N learned on dev (†) also improve performance on the test data. Max ρ indicates correlation at the values of N that lead to the highest Spearman correlation on the development data. For each weighting scheme these are: 140 (tTest), 240 (PPMI), and 20 (Freq). Full ρ indicates the correlation when using full vectors without CS.

The cosine, Jaccard, and Lin similarity measures (Curran, 2004) were all used to ensure the results reflect genuine effects of context selection, and not an artefact of any particular similarity measure. The similarity measure and value of N were chosen, given a particular weighting scheme, to maximise correlation on the development part of the MEN data (Bruni et al., 2012) (**MENdev**). Testing was performed on the remaining section of MEN and the entire WS353 dataset (Finkelstein et al., 2002). The MEN dataset consists of 3,000 word pairs rated for similarity, which is divided into a 2,000-pair development set and a 1,000-pair test set. WS353 consists only of 353 pairs, but has been consistently used as a benchmark word similarity dataset throughout the past decade.

Results Figure 2 shows how correlation varies with N for the MEN development data. The peak performance for tTest is achieved when using around 140 top-ranked contexts per word, while for PPMI it is at $N = 240$, and for Freq $N = 20$. The dramatic drop in performance is demonstrated when using all three similarity measures, although Jaccard seems particularly sensitive to the negative tTest weights that are introduced when lower-ranked contexts are added to the vectors. The remaining experiments only consider cosine similarity. We also find that context selection improves correlation for tTest, PPMI, and the unweighted Freq vectors on the test data (Table 1). Moreover, the lower the correlation from the full vectors, the larger the improvement when using CS.

3.1 Dimensionality Reduction

Figure 3 shows the effects of dimensionality reduction described in the following experiments.

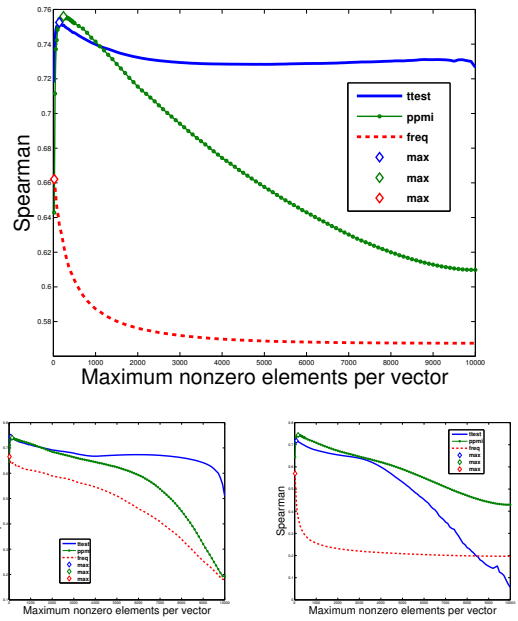

Figure 2: Correlation decreases as more lower-ranked context words are introduced (MENdev), with cosine (top), Lin (bottom left), and Jaccard (bottom right) simialrity measures.

3.1.1 SVD and CS

To check whether CS improves the correlation through increased sparsity or whether it improves the contextual representation of the words, we investigated the behaviour of SVD on three different levels of vector sparsity. To construct the most sparse vectors, we chose the best performing N for each weighting scheme (from Table 1). Thus sparse tTest vectors had $\frac{140}{10000} = 0.0140$, or 1.4%, non-zero elements. We also chose a mid-range of $N = 3300$ for up to 33% of non-zero elements per vector, and finally the full vectors with $N = 10000$.

Results In general the CS-tuned vectors lead to better lower-dimensional representations. The mid-range contexts in the tTest weighting scheme seem to hold information that hinders SVD, while the lowest-ranked negative weights appear to help (when the mid-range contexts are present as well). For the PPMI weighting, fewer contexts consistently lead to better representations, while the unweighted vectors seem to mainly hold information in the top 20 most frequent contexts for each word.

3.1.2 SVD, CS, and Normalisation

We also consider the combination of normalisation and context selection followed by SVD.

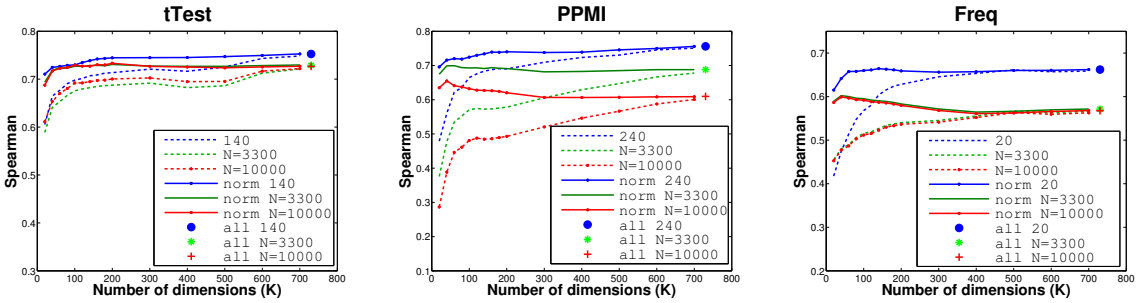

Figure 3: Vectors tuned for sparseness (blue) consistently produce equal or better dimensionality reductions (results on MENdev). The solid lines show improvement in lower dimensional representations of SVD when dimensionality reduction is applied after normalisation.

Results Normalisation leads to more stable SVD representations, with a large improvement for small numbers of dimensions (K) as demonstrated by the solid lines in Figure 3. At $K = 20$ the Spearman correlation increases from 0.61 to 0.71. In addition, for tTest there is an improvement in the mid-range vectors, and a knock-on effect for the full vectors. As the tTest values effectively range from -0.1 to 0.1, the mid-range values are very small numbers closely grouped around zero. Normalisation spreads and increases these numbers, perhaps making them more relevant to the SVD algorithm. The effect is also visible for PPMI weighting where at $K = 20$ the correlation increases from 0.48 to 0.70. For PPMI and Freq we also see that, for the full and mid-range vectors, the SVD representations have slightly higher correlations than the unreduced vectors.

3.2 Random Indexing

We use random indexing primarily to produce a vector representation for convolution (Section 4). While this produces a lower-dimensional representation, it may not use less memory since the resulting vectors, although smaller, are fully dense.

In summary, the RI encoded vectors with dimensions of $D = 4096$ lead to only slightly reduced correlation values compared to their unencoded counterparts. We find that for tTest we get similar performance with or without CS at any level, while for PPMI CS helps especially for $D \geqslant 512$. On Freq we find that CS with $N = 60$ leads to higher correlation, but mid-range and full vectors have equivalent performance. For Freq, the correlation is equivalent to full vectors from $D = 128$, while for the weighted vectors 512 dimensions appear to be sufficient. Unlike for SVD, normalisation slightly reduces the performance for mid-range dimensions.

4 Compositional Experiments

We examine the performance of vectors augmented by CS and normalisation in two compositional tasks. The first is an extension of the word similarity task to phrase pairs, using the dataset of Mitchell and Lapata (2010). Each entry in the dataset consists of two phrases, each consisting of two words (in various syntactic relations, such as verb-object and adjective noun), and a gold standard score. We combine the two word vectors into a single phrase vector using various operators described below. We then calculate the similarity between the phrase vectors using cosine and compare the resulting scores against the gold standard using Spearman correlation. The second task is our new definitions task where, again, word vectors from each definition are composed to form a single vector, which can then be compared for similarity with the target term.

We use PPMI- and tTest-weighted vectors at three CS cutoff points: the best chosen N from Section 3, the top third of the ranked contexts at $N = 3300$, and the full vectors without CS at $N = 10000$. This gives us a range of values to examine, without directly tuning on this dataset. For dimensionality reduction we consider vectors reduced with SVD to 100 and 700 dimensions. In some cases we exclude the results for SVD_{700} because they are very close to the scores for unreduced vectors. We experiment with 3 values of D from $\{512, 1024, 4096\}$ for the RI vectors.

Operators To combine distributional vectors into a single-vector sentence representation, we use a representative set of methods from Mitchell and Lapata (2010). In particular, we use vector addition, elementwise (Hadamard) product, Kronecker product, and circular convolution (Plate, 1991; Jones and Mewhort, 2007), which are de-

fined as follows for two word vectors $\vec{x}, \vec{y}$:

$$\textbf{Sum} \quad \vec{x} + \vec{y} = \{\vec{x}_i + \vec{y}_i\}_i$$
$$\textbf{Prod} \quad \vec{x} \odot \vec{y} = \{\vec{x}_i \cdot \vec{y}_i\}_i$$
$$\textbf{Kron} \quad \vec{x} \otimes \vec{y} = \{\vec{x}_i \cdot \vec{y}_j\}_{ij}$$
$$\textbf{Conv} \quad \vec{x} \circledast \vec{y} = \left\{\sum_{j=0}^{n} (\vec{x})_{j\%n} \cdot (\vec{y})_{(i-j)\%n}\right\}_i$$

Repeated application of the **Sum** operation adds contexts for each of the words that occur in a phrase, which maintains (and mixes) any noisy parts of the component word vectors. Our intention was that use of the CS vectors would lead to less noisy word vectors and hence less noisy phrase and sentence vectors. The **Prod** operator, on the other hand, provides a phrase or sentence representation consisting only of the contexts that are common to all of the words in the sentence (since zeros in any of the word vectors lead to zeros in the same position in the sentence vector). This effect is particularly problematic for rare words which may have sparse vectors, leading to a sparse vector for the sentence.[1] We address the sparsity problem through the use of dimensionality reduction, which produces more dense vectors.

Kron, the Kronecker (or tensor) product of two vectors, produces a matrix (second order tensor) whose diagonal matches the result of the **Prod** operation, but whose off-diagonal entries are all the other products of elements of the two vectors. We only apply **Kron** to SVD-reduced vectors, and to compare two matrices we turn them into vectors by concatenating matrix rows, and use cosine similarity on the resulting vectors. While in the more complex, type-driven methods (Baroni and Zamparelli, 2010; Coecke et al., 2010) tensors represent functions, and off-diagonal entries have a particular transformational interpretation as part of a linear map, the significance of the off-diagonal elements is difficult to interpret in our setting, apart from their role as encoders of the order of operands. We only examine **Kron** as the unencoded version of the **Conv** operator to see how the performance is affected by the random indexing and the modular summation by which **Conv** differs from **Kron**.[2] We cannot use **Kron** for combining more than two words as the size of the resulting tensor grows exponentially with the number of vector operations, but we can use **Conv** as an encoded alternative as it results in a vector of the same dimension as the two operands.

[1] Sparsity is a problem that may be addressable through smoothing (Zhai and Lafferty, 2001), although we do not investigate that avenue in this paper.

[2] **Conv** also differs from **Kron** in that it is commutative, unless one of the operands is permuted. In this paper we do not permute the operands.

Oper		N=140	N=3300	N=10000
sum	ttest	0.40 (**0.41**)	0.40 (0.40)	0.40 (0.40)
	SVD_{100}	0.37 (**0.42**)	0.35 (**0.41**)	0.37 (0.40)
prod	ttest	0.32 (0.32)	**0.40** (**0.40**)	0.32 (0.32)
	SVD_{100}	0.25 (0.23)	0.23 (0.23)	0.21 (0.23)
kron	SVD_{100}	0.31 (0.34)	0.34 (**0.38**)	0.29 (0.32)
	SVD_{700}	**0.39** (**0.39**)	**0.37** (**0.37**)	0.30 (0.30)
conv	RI_{512}	0.10 (0.12)	**0.26** (0.21)	0.25 (0.25)
	RI_{1024}	0.22 (0.15)	0.29 (0.27)	0.25 (0.26)
	RI_{4096}	0.16 (0.19)	**0.33** (**0.34**)	0.28 (0.30)

Table 2: Behaviour of vector operators with tTest vectors on ML2010 (Spearman correlation). Values for normalised vectors in parentheses.

Oper		N=240	N=3300	N=10000
sum	ppmi	**0.40** (0.39)	**0.40** (**0.39**)	0.29 (0.29)
	SVD_{100}	**0.40** (0.40)	0.38 (**0.40**)	0.29 (0.30)
prod	ppmi	0.28 (0.28)	**0.40** (**0.40**)	0.30 (0.30)
	SVD_{100}	0.23 (0.17)	0.18 (0.22)	0.14 (0.12)
kron	SVD_{100}	0.37 (0.30)	0.36 (**0.38**)	0.27 (0.27)
	SVD_{700}	**0.38** (**0.37**)	**0.37** (**0.37**)	0.26 (0.26)
conv	RI_{512}	0.09 (0.09)	0.27 (**0.30**)	0.25 (0.24)
	RI_{1024}	0.08 (0.14)	0.33 (**0.37**)	0.25 (0.27)
	RI_{4096}	0.18 (0.19)	**0.37** (**0.38**)	0.27 (0.27)

Table 3: Behaviour of vector operators with PPMI vectors on ML2010 (Spearman correlation). Values for normalised vectors in parentheses.

4.1 Phrase Similarity

To test how CS, normalisation, and dimensionality reduction affect simple compositional vector operations we use the test portion of the phrasal similarity dataset from Mitchell and Lapata (2010) (ML2010). This dataset consists of pairs of two-word phrases and a human similarity judgement on the scale of 1-7. There are three types of phrases: noun-noun, adjective-noun, and verb-object. In the original paper, and some subsequent works, these were treated as three different datasets; however, here we combine the datasets into one single phrase pair dataset. This allows us to summarise the effects of different types of vectors on phrasal composition in general.

Results Our results (Tables 2 and 3) are comparable to those in Mitchell and Lapata (2010) averaged across the phrase-types ($\rho = 0.44$), but are achieved with much smaller vectors. We find that with normalisation, and the optimal choice of N, there is little difference between **Prod** and **Sum**. **Sum** and **Kron** benefit from normalisation, especially in combination with SVD, but for **Prod** it either makes no difference or reduces performance. Product-based methods (**Prod, Kron,**

Conv) have a preference for context selection that includes the mid-rank contexts ($N = 3300$), but not the full vector ($N = 10000$). On tTest vectors **Sum** is relatively stable across different CS and SVD settings, but with PPMI weighting, there is a preference for lower N. SVD reduces performance for **Prod**, but not for **Kron**. Finally, **Conv** gets higher correlation with higher-dimensional RI vectors and with PPMI weights.

4.2 Definition Retrieval

In this task, which is formulated as a retrieval task, we investigate the behaviour of different vector operators as multiple operations are chained together. We first encode each definition into a single vector through repeated application of one of the operators on the distributional vectors of the content words in the definition. Then, for each head (defined) word, we rank all the different definition vectors in decreasing order according to inner product (unnormalised cosine) similarity with the head word's distributional vector.

Performance is measured using precision and Mean Reciprocal Rank (MRR). If the correct definition is ranked first, the precision (P@1) is 1, otherwise 0. Since there is only one definition per head word, the reciprocal rank (RR) is the inverse of the rank of the correct definition. So if the correct definition is ranked fourth, for example, then RR is $\frac{1}{4}$. MRR is the average of the RR across all head words.

The difficulty of the task depends on how many words there are in the dataset and how similar their definitions are. In addition, if a head word occurs in the definition of another word in the same dataset, it may cause the incorrect definition to be ranked higher than the correct one. These problems are more likely to occur with higher frequency words and in a larger dataset. In order to counter these effects, we average our results over ten repeated random samplings of 100 word-definition pairs. The sampling also gives us a random **baseline** for P@1 of 0.0130 ± 0.0106 and for MRR 0.0576 ± 0.0170, which can be interpreted as there is a chance of slightly more than 1 in 100 of ranking the correct definition first, and on average the correct definition is ranked around the 20 mark.

For this task all experiments were performed using the tTest-weighted vectors. When applying normalisation we use $\lambda = 1$ (**Norm**) and $\lambda = 10$ (**Norm10**). In addition, we examine the effect of continually applying **Norm** after every operation (**CNorm**).

DD2	DD3	DD4	DD5	DD6	DD7	DD8	DD9	DD10
346	547	594	537	409	300	216	150	287

Table 4: Number of definitions per dataset.

Dataset We developed a new dataset (**DD**) consisting of 3,386 definitions from the Wiktionary BNC spoken-word frequency list.[3] Most of the words have several definitions, but we considered only the first definition with at least two non-stopwords. The word-definition pairs were divided into nine separate datasets according to the number of non-stopwords in the definition. For example, all of the definitions that have five content words are in **DD5**. The exception is **DD10**, which contains all the definitions of ten or more words. Table 4 shows the number of definitions in each dataset.

Results Figure 4 shows how the MRR varies with different **DD** datasets for **Sum**, **Prod**, and **Conv**. The CS, SVD, and RI settings for each operator correspond to the best average settings from Table 5. In some cases other settings had similar performance, but we chose these for illustrative purposes. We can see that all operators have relatively higher MRR on smaller datasets (**DD6-9**). Compensating for that effect, we can hypothesise that **Sum** has a steady performance across different definition sizes, while the performance of both **Prod** and **Conv** declines as the number of operations increases. Normalisation helps with **Sum** throughout, with little difference in performance between **Norm** and **Norm10**, but with a slight decrease when **CNorm** is used. On the other hand, only **CNorm** improves the ranking of **Prod**-based vectors. Normalisation makes no difference for RI vectors combined with convolution and the results in Table 5 show that, on average, **Conv** performs worse than the random baseline.

In Figure 5 we can see that, although dimensionality reduction leads to lower MRR, for **Sum**, normalisation prior to SVD counteracts this effect, while, for **Prod**, dimensionality reduction, in general, reduces the performance.

[3] http://simple.wiktionary.org/wiki/Wiktionary:BNC_spoken_freq

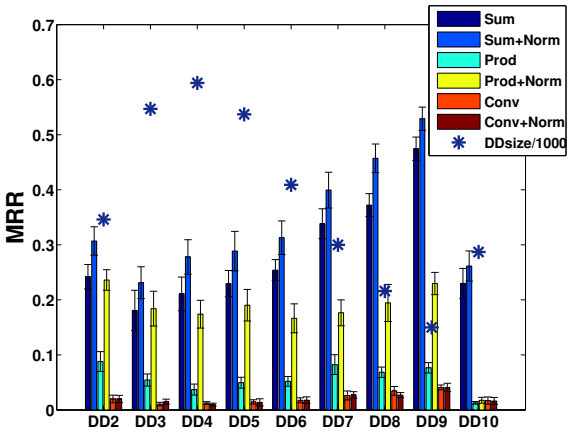

Figure 4: Per-dataset breakdown of best normalised and unnormalised vectors for each vector operator. Stars indicate the dataset size from Table 4 divided by 1000.

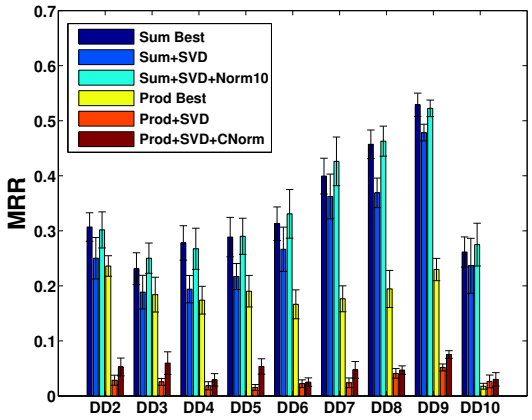

Figure 5: Per-dataset breakdown of best normalised and unnormalised SVD vectors for **Sum** and **Prod**. For both operators the best CS and SVD settings for normalised vectors were $N = 140$, $K = 700$, and for unnormalised were $N = 10000$, $K = 700$.

	Sum		Prod		Conv	
Norm	**No**	**Yes**	**No**	**CN**	**No**	**Yes**
CS (N)	140	140	3300	10000	140	3300
SVD(K)**/RI**(D)	700	700	None	None	2048	512
mean P@1	0.18	**0.23**	0.01	0.11	0.00	0.00
mean MRR	0.28	**0.35**	0.06	0.17	0.02	0.02

Table 5: Best settings for operators calculated from the highest average MRR across all the datasets, with and without normalisation. The results for vectors with no normalisation or CS are: **Sum** - P@1=0.1567, MRR=0.2624; **Prod** - P@1=0.0147, MRR=0.0542; **Conv** P@1=0.0027, MRR=0.0192.

5 Discussion

In this paper we introduced context selection and normalisation as techniques for improving the semantic vector space representations of words. We found that, although our untuned vectors perform better on WS353 data ($\rho = 0.63$) than vectors used by Mitchell and Lapata (2010) ($\rho = 0.42$), our best phrase composition model (**Sum**, $\rho = 0.40$) produces a lower performance than an estimate of their best model (**Prod**, $\rho = 0.44$).[4] This indicates that better performance on word-similarity data does not directly translate into better performance on compositional tasks; however, CS and normalisation are both effective in increasing the quality of the composed representation ($\rho = 0.42$). Since CS and normalisation are computationally inexpensive, they are an excellent way to improve model quality compared to the alternative, which is building several models with various context types, in order to find which one suits the data best.

Furthermore, we show that, as the number of vector operations increases, **Sum** is the most stable operator and that it benefits from sparser representations (low N) and normalisation. Employing both of these methods, we are able to build an SVD-based representation that performs as well as full-dimensional vectors which, together with **Sum**, give the best results on both phrase and definition tasks. In fact, normalisation and CS both improve the SVD representations of the vectors across different weighting schemes. This is a key result, as many of the more complex compositional methods require low dimensional representations for computational reasons.

Future work will include application of CS and normalised lower-dimensional vectors to more complex compositional methods, and investigations into whether these strategies apply to other context types and other dimensionality reduction methods such as LDA (Blei et al., 2003).

[4]The estimate is computed as an average across the three phrase-type results.

Acknowledgements

Tamara Polajnar is supported by ERC Starting Grant DisCoTex (306920). Stephen Clark is supported by ERC Starting Grant DisCoTex (306920) and EPSRC grant EP/I037512/1. We would like to thank Laura Rimell for helpful discussion, and Laura and the anonymous reviewers for helpful comments on the paper.

References

Jacob Andreas and Zoubin Ghahramani. 2013. A generative model of vector space semantics. In *Proceedings of the ACL 2013 Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

M. Baroni and R. Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, Cambridge, MA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea, July. Association for Computational Linguistics.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Comput. Linguist.*, 38(1):41–71, March.

B. Coecke, M. Sadrzadeh, and S. Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In J. van Bentham, M. Moortgat, and W. Buszkowski, editors, *Linguistic Analysis (Lambek Festschrift)*, volume 36, pages 345–384.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Scott Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20:116–131.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Pablo Gamallo and Stefan Bordag. 2011. Is singular value decomposition useful for word similarity extraction? *Language Resources and Evaluation*, 45(2):95–119.

Michael N. Jones and Douglas J. K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

Ekaterina Kochmar and Ted Briscoe. 2013. Capturing anomalies in the choice of content words in compositional distributional semantic space. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP-2013)*, Hissar, Bulgaria.

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(8):1388–1429.

T. A. Plate. 1991. Holographic reduced Representations: Convolution algebra for compositional distributed representations. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, August 1991*, pages 30–35, San Mateo, CA. Morgan Kauffman.

D Seung and L Lee. 2001. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562.

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)*, Jeju Island, Korea.

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

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2011. Distributed structures and distributional meaning. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, DiSCo-11, pages 10–15, Portland, Oregon. Association for Computational Linguistics.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 334–342, New York, NY, USA. ACM.