Reducing Dimensions of Tensors in Type-Driven Distributional Semantics

Tamara Polajnar Luana Făgărăşan Stephen Clark
Computer Laboratory
University of Cambridge
Cambridge, UK
first.last@cl.cam.ac.uk

Abstract

Compositional distributional semantics is a subfield of Computational Linguistics which investigates methods for representing the meanings of phrases and sentences. In this paper, we explore implementations of a framework based on Combinatory Categorial Grammar (CCG), in which words with certain grammatical types have meanings represented by multilinear maps (i.e. multi-dimensional arrays, or tensors). An obstacle to full implementation of the framework is the size of these tensors. We examine the performance of lower dimensional approximations of transitive verb tensors on a sentence plausibility/selectional preference task. We find that the matrices perform as well as, and sometimes even better than, full tensors, allowing a reduction in the number of parameters needed to model the framework.

1 Introduction

An emerging subfield of computational linguistics is concerned with learning compositional distributional representations of meaning (Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010; Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Clarke, 2012; Socher et al., 2012; Clark, 2013). The advantage of such representations lies in their potential to combine the benefits of distributional approachs to word meaning (Schütze, 1998; Turney and Pantel, 2010) with the more traditional compositional methods from formal semantics (Dowty et al., 1981). Distributional representations have the properties of robustness, learnability from data, ease of handling ambiguity, and the ability to represent gradations of meaning; whereas compositional models handle the unbounded nature of natural language, as well as providing established accounts of logical words, quantification, and inference.

One promising approach which attempts to combine elements of compositional and distributional semantics is by Coecke et al. (2010). The underlying idea is to take the type-driven approach from formal semantics — in particular the idea that the meanings of complex grammatical types should be represented as functions — and apply it to distributional representations. Since the mathematics of distributional semantics is provided by linear algebra, a natural set of functions to consider is the set of linear maps. Coecke et al. recognize that there is a natural correspondence from complex grammatical types to tensors (multi-linear maps), so that the meaning of an adjective, for example, is represented by a matrix (a 2nd-order tensor)[1] and the meaning of a transitive verb is represented by a 3rd-order tensor.

Coecke et al. use the grammar of pregroups as the syntactic machinery to construct distributional meaning representations, since both pregroups and vector spaces can be seen as examples of the same abstract structure, which leads to a particularly clean mathematical description of the compositional process. However, the approach applies more generally, for example to other forms of categorial grammar, such as Combinatory Categorial Grammar (Steedman, 2000; Maillard et al., 2014), and also to phrase-structure grammars in a way that a formal linguist would recognize (Baroni et al., 2014). Clark (2013) provides a description of the tensor-based framework aimed more at computational linguists, relying only on the mathematics of multi-linear algebra rather than the category theory used in Coecke et al. (2010). Section 2 repeats some of this description.

A major open question associated with the tensor-based semantic framework is how to learn

[1] This same insight lies behind the work of Baroni and Zamparelli (2010).

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1036–1046,
October 25-29, 2014, Doha, Qatar.

the tensors representing the meanings of words with complex types, such as verbs and adjectives. The framework is essentially a compositional framework, providing a recipe for how to combine distributional representations, but leaving open what the underlying vector spaces are and how they can be acquired. One significant challenge is an engineering one: in a wide-coverage grammar, which is able to handle naturally occurring text, there will be a) a large lexicon with many word-category pairs requiring tensor representations; and b) many higher-order tensors with large numbers of parameters which need to be learned. In this paper we take a first step towards learning such representations, by learning tensors for transitive verbs.

One feature of the tensor-based framework is that it allows the meanings of words and phrases with different basic types, for example nouns and sentences, to live in different vector spaces. This means that the sentence space is task specific, and must be defined in advance. For example, to calculate sentence similarity, we would have to learn a vector space where distances between vectors representing the meanings of sentences reflect similarity scores assigned by human annotators.

In this paper we describe an initial investigation into the learning of word meanings with complex syntactic types, together with a simple sentence space. The space we consider is the "plausibility space" described by Clark (2013), together with sentences of the form subject-verb-object. This space is defined to distinguish semantically plausible sentences (e.g. *Animals eat plants*) from implausible ones (e.g. *Animals eat planets*). Plausibility can be either represented as a single continuous variable between 0 and 1, or as a two-dimensional probability distribution over the classes *plausible* ($\top$) and *implausible* ($\bot$). Whether we consider a one- or two-dimensional sentence space depends on the architecture of the logistic regression classifier that is used to learn the verb (Section 3).

We begin with this simple plausibility sentence space to determine if, in fact, the tensor-based representation can be learned to a sufficiently useful degree. Other simple sentence spaces which can perhaps be represented using one or two variables include a "sentence space" for the sentiment analysis task (Socher et al., 2013), where one variable represents positive sentiment and the other negative. We also expect that the insights gained from research on this task can be applied to more complex sentence spaces, for example a semantic similarity space which will require more than two variables.

2 Syntactic Types to Tensors

The syntactic type of a transitive verb in English is $(S\backslash NP)/NP$ (using notation from Steedman (2000)), meaning that a transitive verb is a function which takes an NP argument to the right, an NP argument to the left, and results in a sentence S. Such *categories* with slashes are *complex categories*; S and NP are *basic* or *atomic* categories. Interpreting such categories under the Coecke et al. framework is straightforward. First, for each atomic category there is a corresponding vector space; in this case the sentence space $\mathbf{S}$ and the noun space $\mathbf{N}$.[2] Hence the meaning of a noun or noun phrase, for example *people*, will be a vector in the noun space: $\overrightarrow{people} \in \mathbf{N}$. In order to obtain the meaning of a transitive verb, each slash is replaced with a tensor product operator, so that the meaning of *eat*, for example, is a 3rd-order tensor: $\overline{eat} \in \mathbf{S} \otimes \mathbf{N} \otimes \mathbf{N}$. Just as in the syntactic case, the meaning of a transitive verb is a function (a multi-linear map) which takes two noun vectors as arguments and returns a sentence vector.

Meanings combine using *tensor contraction*, which can be thought of as a multi-linear generalisation of matrix multiplication (Grefenstette, 2013). Consider first the adjective-noun case, for example *black cat*. The syntactic type of *black* is N/N; hence its meaning is a 2nd-order tensor (matrix): $\overline{black} \in \mathbf{N} \otimes \mathbf{N}$. In the syntax, N/N combines with N using the rule of forward application ($N/N \ N \Rightarrow N$), which is an instance of function application. Function application is also used in the tensor-based semantics, which, for a matrix and vector argument, corresponds to matrix multiplication.

Figure 1 shows how the syntactic types combine with a transitive verb, and the corresponding tensor-based semantic types. Note that, after the verb has combined with its object NP, the type of the verb phrase is $S\backslash NP$, with a corresponding meaning tensor (matrix) in $\mathbf{S} \otimes \mathbf{N}$. This matrix then combines with the subject vector, through

[2] In practice, for example using the CCG parser of Clark and Curran (2007), there will be additional atomic categories, such as PP, but not many more.

$$
\begin{array}{ccc}
\textit{people} & \textit{eat} & \textit{fish} \\
\overline{NP} & \overline{(S\backslash NP)/NP} & \overline{NP} \\
\mathbf{N} & \mathbf{S}\otimes\mathbf{N}\otimes\mathbf{N} & \mathbf{N} \\
 & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad}{>}} \\
 & \multicolumn{2}{c}{S\backslash NP} \\
 & \multicolumn{2}{c}{\mathbf{S}\otimes\mathbf{N}} \\
\multicolumn{3}{c}{\overline{\qquad\qquad\qquad\qquad\qquad}{<}} \\
\multicolumn{3}{c}{S} \\
\multicolumn{3}{c}{\mathbf{S}}
\end{array}
$$

Figure 1: Syntactic reduction and tensor-based semantic types for a transitive verb sentence

matrix multiplication, to give a sentence vector.

In practice, using for example the wide-coverage grammar from CCGbank (Hockenmaier and Steedman, 2007), there will be many types with more than 3 slashes, with corresponding higher-order tensors. For example, a common category for a preposition is the following: $((S\backslash NP)\backslash(S\backslash NP))/NP$, which would be assigned to WITH in *eat* WITH *a fork*. (The way to read the syntactic type is as follows: *with* requires an NP argument to the right – *a fork* in this example – and then a verb phrase to the left – *eat* with type $S\backslash NP$ – resulting in a verb phrase $S\backslash NP$.) The corresponding meaning tensor lives in the tensor space $\mathbf{S}\otimes\mathbf{N}\otimes\mathbf{S}\otimes\mathbf{N}\otimes\mathbf{N}$, i.e. a 5th-order tensor. Categories with even more slashes are not uncommon, for example $((N/N)/(N/N))/((N/N)/(N/N))$. Clearly learning parameters for such tensors is highly challenging, and it is likely that lower dimensional approximations will be required.

3 Methods

In this paper we compare five different methods for modelling the type-driven semantic representation of subject-verb-object sentences. The tensor is a function that encodes the meaning of a verb. It takes two vectors from the K-dimensional noun space as input, and produces a representation of the sentence in the S-dimensional sentence space. In this paper, we consider a *plausibility space* where $\mathbf{S}$ is either a single variable or a two-dimensional space over two classes: *plausible* ($\top$) and *implausible* ($\bot$).

The first method (**Tensor**) follows Krishnamurthy and Mitchell (2013) by learning a tensor as parameters in a softmax classifier. We introduce three related methods (**2Mat**, **SKMat**, **KKMat**), all of which model the verb as a matrix or a pair of matrices (Figure 2). Table 1 gives the number of

	Tensor	2Mat	SKMat	KKMat	DMat
V	$2K^2$	$4K$	$2K$	K^2	K^2
Θ	4	8	4	0	0

Table 1: Number of parameters per method.

parameters for each method. **Tensor**, **2Mat**, and **SKMat** all have a two-dimensional $\mathbf{S}$ space, while **KKMat** produces a scalar value. In all of these learning-based methods the derivatives were obtained via the chain rule with respect to each set of parameters and gradient descent was performed using the Adagrad algorithm (Duchi et al., 2011).

We also reimplement a distributional method (**DMat**), which was previously used in SVO experiments with the type-driven framework (Grefenstette and Sadrzadeh, 2011). While the other methods are trained as plausibility classifiers, in **DMat** we estimate the class boundary from cosine similarity via training data (see explanation below).

Tensor If subject (n_s) and object (n_o) nouns are K-dimensional vectors and the plausibility vector is S-dimensional with $S = 2$, we can learn the values of the $K \times K \times S$ tensor representing the verb as parameters (V) of a regression algorithm. To represent this space as a distribution over two classes ($\top$,$\bot$) we apply a sigmoid function (σ) to restrict the output to the [0,1] range and the softmax activation function (g) to balance the class probabilities. The full parameter set which needs to be optimised for is $\mathrm{B} = \{\mathrm{V}, \Theta\}$, where $\Theta = \{\theta_\top, \theta_\bot\}$ are the softmax parameters for the two classes. For each verb we optimise the KL-divergence $\mathcal{L}$ between the training labels t^i and classifier predictions using the following regularised objective:

$$O(\mathrm{B}) = \sum_{i=1}^{N} \mathcal{L}\left(t^i, g\left(\sigma\left(h_{\mathrm{V}}\left(n_s^i, n_o^i\right)\right), \Theta\right)\right) + \frac{\lambda}{2}||\mathrm{B}||^2 \quad (1)$$

where n_s^i and n_o^i are the subject and object of the training instance $i \in N$, and $h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = (n_s^i)\mathrm{V}(n_o^i)^T$ describes tensor contraction. The function h_{V} is described diagrammatically in Figure 2-**(a)**, where the verb tensor parameters are drawn as a cube with the subject and object noun vectors as operands on either side. The output is a two-dimensional vector which is then transformed using the sigmoid and softmax functions.

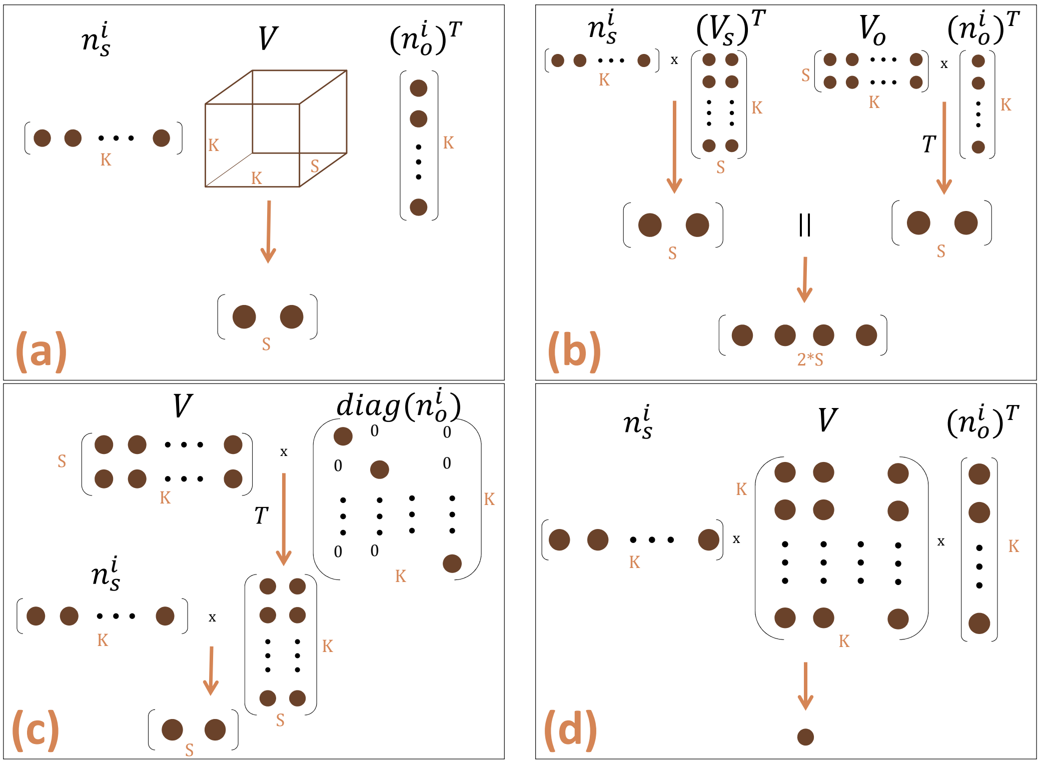

Figure 2: Illustrations of the h_{V} function for the regression-based methods **(a)-Tensor**, **(b)-2Mat**, **(c)-SKMat**, **(d)-KKMat**. The operation in **(a)** is tensor contraction, T denotes transpose, and $\times$ denotes matrix multiplication.

The gold-standard distribution over training labels is defined as $(1,0)$ or $(0,1)$, depending on whether the training instance is a positive (plausible) or negative (implausible) example. Tensor contraction is implemented using the Matlab Tensor Toolbox (Bader et al., 2012).

2Mat An alternative approach is to decouple the interaction between the object and subject by learning a pair of $S \times K$ ($S = 2$) matrices (V_s, V_o) for each of the input noun vectors (one matrix for the subject slot of the verb and one for the object slot). The resulting S-vectors are concatenated, after the subject and object nouns have been combined with their matrices, and combined with the softmax component to produce the output distribution. Therefore the objective function is the same as in Equation 1, but h_{V} is defined as:

$$h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = \left((n_s^i)\mathrm{V}_s^T\right) \,||\, \left(\mathrm{V}_o(n_o^i)^T\right)^T$$

where $||$ represents vector concatenation. The intention is to test whether we can learn the verb without directly multiplying subject and object features, n_s^i and n_o^j. The function h_{V} is shown in Figure 2-**(b)**, where the verb tensor parameters are drawn as two $2 \times K$ matrices, one of which interacts with the subject and the other with the object noun vector. The output is a four-dimensional vector whose values are then restricted to [0,1] using the sigmoid function and then transformed into a two-dimensional distribution over the classes using the softmax function.

SKMat A third option for generating a sentence vector with $S = 2$ dimensions is to consider the verb as an $S \times K$ matrix. If we transform the object vector into a $K \times K$ matrix with the noun on the diagonal and zeroes elsewhere, we can combine the verb and object to produce a new $S \times K$ matrix, which is encoding the meaning of the verb phrase. We can then complete the sentence reduction by multiplying the subject vector with this verb phrase vector to produce an S-dimensional sentence vector. Formally, we define **SKMat** as:

$$h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = n_s^i\left(\mathrm{V}diag(n_o^i)\right)^T$$

and use it in Equation 1. The function h_{V} is described in Figure 2-**(c)**, where the verb tensor parameters are drawn as a matrix, the subject as a vector, and the object as a diagonal matrix. The graphic demonstrates the two-step combination and the intermediate $S \times K$ verb phrase matrix, as well as the the noun vector product that results in a two-dimensional vector which is then transformed using the sigmoid and softmax functions. Whilst the tensor method captures the interactions between all pairs of context features $(n_{si} \cdot n_{oj})$, **SKMat** only captures the interactions between matching features $(n_{si} \cdot n_{oi})$.

KKMat Given a two-class problem, such as plausibility classification, the softmax implementation is overparameterised because the class membership can be estimated with a single variable. To produce a scalar output, we can learn the parameters for a single $K \times K$ matrix (V) using standard logistic regression with the mean squared error cost function:

$$O(\mathrm{V}) = -\frac{1}{m}\left[\sum_{i=1}^{N} t^i \log h_{\mathrm{V}}\left(n_s^i, n_o^i\right) + (1-t^i)\log h_{\mathrm{V}}\left(n_s^i, n_o^i\right)\right]$$

where $h_{\mathrm{V}}\left(n_s^i, n_o^i\right) = (n_s^i)\mathrm{V}(n_o^i)^T$ and the objective is regularised: $O(\mathrm{V}) + \frac{\lambda}{2}||\mathrm{V}||^2$. This function is shown in Figure 2-**(d)**, where the verb parameters are shown as a matrix, while the subject and object are vectors. The output is a single scalar, which is then transformed with the sigmoid function. Values over 0.5 are considered plausible.

DMat The final method produces a scalar as in **KKMat**, but is distributional and based on corpus counts rather than regression-based. Grefenstette and Sadrzadeh (2011) introduced a corpus-based approach for generating a $K \times K$ matrix for each verb from an average of Kronecker products of the subject and object vectors from the positively labelled subset of the training data. The intuition is that, for example, the matrix for *eat* may have a high value for the contextual topic pair describing animate subjects and edible objects. To determine the plausibility of a new subject-object pair for a particular verb, we calculate the Kronecker product of the subject and object noun vectors for this pair, and compare the resulting matrix with the average verb matrix using cosine similarity.

For label prediction, we calculate the similarity between each of the training data pairs and the learned average matrix. Unlike for **KKmat**, the **class cutoff** is estimated at the break-even point of the receiver operator characteristic (ROC) generated by comparing the training labels with this cosine similarity value. The break-even point is when the true positive rate is equal to the false positive rate. In practice it would be more accurate to estimate the cutoff on a validation dataset, but some of the verbs have so few training instances that this was not possible.

4 Experiments

In order to examine the quality of learning we run several experiments where we compare the different methods. In these experiments we consider the **DMat** method as the baseline. Some of the experiments employ cross-validation, in particular five repetitions of 2-fold cross validation (5x2cv), which has been shown to be statistically more robust than the traditional 10-fold cross validation (Alpaydin, 1999; Ulaş et al., 2012). The results of 5x2cv experiments can be compared using the regular paired t-test, but the specially designed 5x2cv F-test has been proven to produce fewer statistical errors (Ulaş et al., 2012).

The performance was evaluated using the area under the ROC (AUC) and the F_1 measure (based on precision and recall over the plausible class). The AUC evaluates whether a method is ranking positive examples above negative ones, regardless of the class cutoff value. F_1 shows how accurately a method assigns the correct class label. Another way to interpret the results is to consider the AUC as the measure of the quality of the parameters in the verb matrix or tensor, while the F-score indicates how well the softmax, the sigmoid, and the **DMat** cutoff algorithm are estimating class participation.

Ex-1. In the first experiment, we compare the different transitive verb representations by running 5x2cv experiments on ten verbs chosen to cover a range of concreteness and frequency values (Section 4.2).

Ex-2. In the initial experiments we found that some models had low performance, so we applied the column normalisation technique, which is often used with regression learning to standardise the numerical range of features:

$$\vec{x} := \frac{\vec{x} - \min(\vec{x})}{\max(\vec{x}) - \min(\vec{x})} \tag{2}$$

This preserves the relative values of features between training samples, while moving the values to the [0,1] range.

Ex-3. There are varying numbers of training examples for each of the verbs, so we repeated the 5x2cv with datasets of 52 training points for each verb, since this is the size of the smallest dataset of the verb CENSOR. The points were randomly sampled from the datasets used in the first experiment. Finally, the four verbs with the largest datasets were used to examine how the performance of the methods changes as the amount of training data increases. The 4,000 training samples were randomised and half were used for testing. We sampled between 10 and 1000 training triples from the other half (Figure 4).

Verb	Concreteness	# of Positive	Frequency
APPLY	2.5	5618	47361762
CENSOR	3	26	278525
COMB	5	164	644447
DEPOSE	2.5	118	874463
EAT	4.44	5067	26396728
IDEALIZE	1.17	99	485580
INCUBATE	3.5	82	833621
JUSTIFY	1.45	5636	10517616
REDUCE	2	26917	40336784
WIPE	4	1090	6348595

Table 2: The 10 chosen verbs together with their concreteness scores. The number of positive SVO examples was capped at 2000. **Frequency** is the frequency of the verb in the GSN corpus.

4.1 Noun vectors

Distributional semantic models (Turney and Pantel, 2010) encode word meaning in a vector format by counting co-occurrences with other words within a specified context window. We constructed the vectors from the October 2013 dump of Wikipedia articles, which was tokenised using the Stanford NLP tools[3], lemmatised with the Morpha lemmatiser (Minnen et al., 2001), and parsed with the C&C parser (Clark and Curran, 2007). In this paper we use sentence boundaries to define context windows and the top 10,000 most frequent lemmatised words in the whole corpus (excluding stopwords) as context words. The raw co-occurrence counts are re-weighted using the standard tTest weighting scheme (Curran, 2004), where $f_{w_i c_j}$ is the number of times target noun w_i occurs with context word c_j:

$$tTest(\vec{w_i}, c_j) = \frac{p(w_i, c_j) - p(w_i)p(c_j)}{\sqrt{p(w_i)p(c_j)}} \quad (3)$$

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

Using all 10,000 context words would result in a large number of parameters for each verb tensor, and so we apply singular value decomposition (SVD) (Turney and Pantel, 2010) with 40 latent dimensions to the target-context word matrix. We use context selection (with $N = 140$) and row normalisation as described in Polajnar and Clark (2014) to markedly improve the performance of SVD on smaller dimensions (K) and enable us to train the verb tensors using very low-dimensional noun vectors. Performance of the noun vectors was measured on standard word similarity datasets and the results were comparable to those reported by Polajnar and Clark (2014).

4.2 Training data

In order to generate training data we made use of two large corpora: the Google Syntactic N-grams (GSN) (Goldberg and Orwant, 2013) and the Wikipedia October 2013 dump. We first chose ten transitive verbs with different concreteness scores (Brysbaert et al., 2013) and frequencies, in order to obtain a variety of verb types. Then the positive (plausible) SVO examples were extracted from the GSN corpus. More precisely, we collected all distinct syntactic trigrams of the form *nsubj ROOT dobj*, where the root of the phrase was one of our target verbs. We lemmatised the words using the NLTK[4] lemmatiser and filtered these examples to retain only the ones that contain nouns that also occur in Wikipedia, obtaining the counts reported in Table 2.

For every positive training example, we constructed a negative (implausible) one by replacing both the subject and the object with a *confounder*, using a standard technique from the selectional preference literature (Chambers and Jurafsky, 2010). A confounder was generated by choosing a random noun from the same frequency bucket as the original noun.[5] Frequency buckets of size 10 were constructed by collecting noun frequency counts from the Wikipedia corpus. For ex-

[3]http://nlp.stanford.edu/software/index.shtml

[4]http://nltk.org/

[5]Note that the random selection of the confounder could result in a plausible negative example by chance, but manual inspection of a subset of the data suggests this happens infrequently for those verbs which select strongly for their arguments, but more often for those verbs that don't.

Verb	Tensor	DMat	KKMat	SKMat	2Mat	Tensor	DMat	KKMat	SKMat	2Mat
	AUC					F_1				
APPLY	85.68†	81.46‡	88.88†‡	68.02	**88.92†‡**	79.27	64.00	**81.24‡**	54.06	80.80‡
CENSOR	79.40	**85.54**	80.55	78.52	83.19	70.66	47.93	**73.52**	37.86	71.07
COMB	89.41	85.65	88.38	69.20†‡	**89.56**	81.15	45.02	81.38	39.67	**82.36**
DEPOSE	92.70	**94.44**	93.12	84.47†	93.20	84.60	54.77	84.79	43.79	**86.15**
EAT	94.62	93.81	95.17	67.92	**95.88‡**	88.91	52.45	88.83	56.22	**89.95**
IDEALIZE	69.56	**75.84**	72.46	61.19	70.23	66.53	48.28	**68.39**	31.03	67.43
INCUBATE	89.33	85.53	88.61	70.59	**91.40**	80.30	50.84	80.90	31.99	**84.55**
JUSTIFY	85.27†	88.70‡	89.97‡	73.56	**90.10‡**	79.73	73.71	81.10	54.09	**82.52**
REDUCE	96.13	95.48	96.69†	79.32	**97.21**	91.24	71.24‡	87.46	76.67‡	**92.22**
WIPE	85.19	84.47	**87.84†**	64.93†‡	81.29	78.57	47.62	80.65	39.50	**78.90**
MEAN	86.93	87.29	**88.37**	71.96	88.30	80.30	55.79	81.03	46.69	**81.79**

Table 3: The best AUC and F_1 results for all the verbs, where † denotes statistical significance compared to **DMat** and ‡ denotes significance compared to **Tensor** according to the 5x2cv F-test with $p < 0.05$.

ample, for the plausible triple *animal* EAT *plant*, we generate the implausible triple *mountain* EAT *product*. Some verbs were well represented in the corpus, so we used up to the top 2,000 most frequent triples for training.

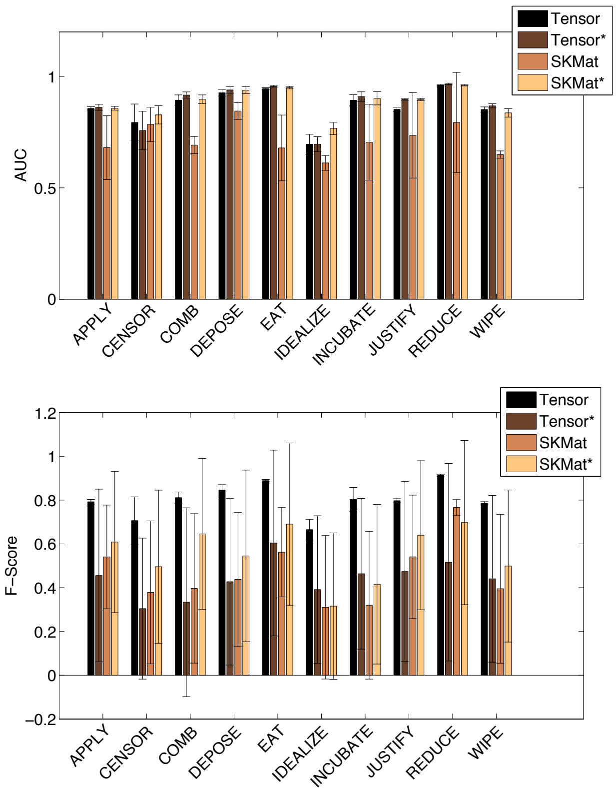

Figure 3: The effect of column normalisation (*) on **Tensor** and **SKMat**. Top table shows AUC and the bottom F_1-score, while the error bars indicate standard deviation.

5 Results

The results from **Ex-1** are summarised in Table 3. We can see that linear regression can lead to models that are able to distinguish between plausible and implausible SVO triples. The **Tensor** method outperforms **DMat**, which was previously shown to produce reasonable verb representations in related experiments (Grefenstette and Sadrzadeh, 2011). **2Mat** and **KKMat**, in turn, outperform **Tensor** demonstrating that it is possible to learn lower dimensional approximations of the tensor-based framework. **2Mat** is an appropriate approximation for functions with two inputs and a sentence space of any dimensionality, while **KKMat** is only appropriate for a single valued sentence space, such as the plausibility or sentiment space. Due to method variance and dataset size there are very few AUC results that are significantly better than **DMat** and even fewer that outperform **Tensor**. All methods perform poorly on the verb IDEALIZE, probably because it has the lowest concreteness value and is in one of the smallest datasets. This verb is also particularly difficult because it does not select strongly for either its subject or object, and so some of the pseudo-negative examples are in fact somewhat plausible (e.g. *town* IDEALIZE *authority* or *child* IDEALIZE *racehorse*). In general, this would indicate that more concrete verbs are easier to learn, as they have a clearer pattern of preferred property types, but there is no distinct correlation.

The results of the normalisation experiments (**Ex-2**) are shown in Table 4. We can see that the **SKMat** method, which performed poorly in **Ex-1** notably improves with normalisation. **Tensor** AUC scores also improve through normalisation, but the F-scores decrease. The rest of the methods, and in particular **DMat** are negatively affected by column normalisation. The results from **Ex-1** and **Ex-2** for **SKMat** and **Tensor** are summarised in

Verb	Tensor	DMat	KKMat	SKMat	2Mat	Tensor	DMat	KKMat	SKMat	2Mat
	AUC					F_1				
APPLY	**86.16**†	48.63‡	82.63†‡	85.73†	85.65†	45.57	46.99	46.17	60.86	**76.60**†
CENSOR	75.74	71.20	78.00	**82.77**	78.64	30.43	55.16	**65.19**	49.59	44.22
COMB	**91.67**†	62.42‡	90.85†	89.79†	91.42†	33.37	61.05	**71.20**	64.56	75.96
DEPOSE	**93.96**†	54.93‡	93.56†	93.87†	93.81†	42.73	39.71	**73.07**	54.51	56.54
EAT	**95.64**†	47.68‡	92.92†	94.99†‡	94.76†	60.42	47.42	58.80	69.05	**87.44**†
IDEALIZE	69.64	55.98	72.20†‡	**76.71**†‡	71.85†	39.14	49.16	41.75	31.57	**50.59**
INCUBATE	**90.97**†	61.31‡	89.69†	90.19†	90.05†	46.35	53.33	**70.45**	41.57	63.61
JUSTIFY	**89.76**†	54.87‡	87.26†‡	89.64†	89.05†	47.38	51.40	41.91	63.96	**80.55**†
REDUCE	**96.63**†	59.58‡	94.99†‡	96.14†	96.53†	51.63	54.27	69.18	69.76	**90.77**†
WIPE	**86.82**†	58.02‡	84.18†	83.65†	86.02†	44.04	55.19	47.84	49.89	**75.80**
MEAN	87.90	57.66	86.83	**88.55**	87.98	44.31	51.57	58.76	55.73	**70.41**

Table 4: The best AUC and F_1 results for all the verbs with normalised vectors, where † denotes statistical significance compared to **DMat** and ‡ denotes significance compared to **Tensor** according to the 5x2cv F-test with $p < 0.05$.

Figure 3. This figure also shows that AUC values have much lower variance, but that high variance in F-score leads to results that are not statistically significant.

When considering the size of the datasets (**Ex-3**), it would seem from Table 5 that **2Mat** is able to learn from less data than **DMat** or **Tensor**. While this may be true over a 5x2cv experiment on small data, Figure 4 shows that this view may be overly simplistic and that different training examples can influence learning. Analysis of errors shows that the baseline method mostly generates false negative errors (i.e. predicting implausible when the gold standard label is plausible). In contrast, **Tensor** produces almost equal numbers of false positives and false negatives, but sometimes produces false negatives with low frequency nouns (e.g. *bourgeoisie* IDEALIZE *work*), presumably because there is not enough information in the noun vector to decide on the correct class. It also produces some false positive errors when either of the nouns is plausible (but the triple is implausible), which would suggest results may be improved by training with data where only one noun is confounded or by treating negative data as possibly positive (Lee and Liu, 2003).

Verb	Tensor	DMat	2Mat
APPLY	**95.76**	86.50	86.31
CENSOR	82.97	**84.09**	77.79
COMB	90.13	92.93	**95.18**
DEPOSE	92.41	91.27	**95.61**
EAT	**99.64**	98.25	99.58
IDEALIZE	75.03	76.68	**88.98**
INCUBATE	91.10	87.20	**96.42**
JUSTIFY	88.96	**88.99**	87.31
REDUCE	**100.0**	99.87	99.46
WIPE	**97.20**	91.63	96.36
MEAN	91.52	89.94	**92.50**

Table 5: Results show average of 5x2cv AUC on small data (26 positive + 26 negative per verb). None of the results are significant.

6 Discussion

Current methods which derive distributed representations for phrases, for example the work of Socher et al. (2012), typically use only matrix representations, and also assume that words, phrases and sentences all live in the same vector space. The tensor-based semantic framework is more flexible, in that it allows different spaces for different grammatical types, which results from it being tied more closely to a type-driven syntactic description; however, this flexibility comes at a cost, since there are many more paramaters to learn.

Various communities are beginning to recognize the additional power that tensor representations can provide, through the capturing of interactions that are difficult to represent with vectors and matrices (see e.g. (Ranzato et al., 2010; Sutskever et al., 2009; Van de Cruys et al., 2012)). Hierarchical recursive structures in language potentially represent a large number of such interactions – the obvious example for this paper being the interaction between a transitive verb's subject and object – and present a significant challenge for machine learning.

This paper is a practical extension of the work in Krishnamurthy and Mitchell (2013), which introduced learning of CCG-based function tensors with logistic regression on a compositional semantics task, but was implemented as a proof-of-concept with vectors of length 2 and on small, manually created datasets based on propositional

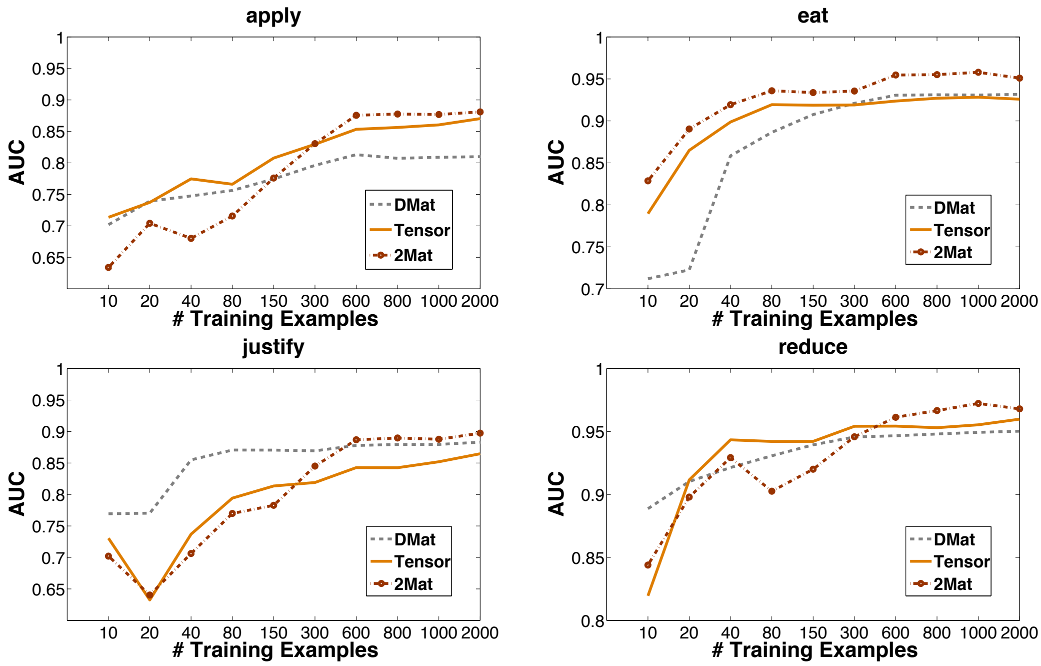

Figure 4: Comparison of **DMat**, **Tensor**, and **2Mat** methods as the number of training instances increases.

logic examples. Here, we go beyond this by learning tensors using corpus data and by deriving several different matrix representations for the verb in the subject-verb-object (SVO) sentence.

This work can also be thought of as applying neural network learning techniques to the classic problem of selectional preference acquisition, since the design of the pseudo-disambiguation experiments is taken from the literature on selectional preferences (Clark and Weir, 2002; Chambers and Jurafsky, 2010). We do not compare directly with methods from this literature, e.g. those based on WordNet (Resnik, 1996; Clark and Weir, 2002) or topic modelling techniques (Seaghdha, 2010), since our goal in this paper is not to extend the state-of-the-art in that area, but rather to use selectional preference acquisition as a test bed for the tensor-based semantic framework.

7 Conclusion

In this paper we introduced three dimensionally reduced representations of the transitive verb tensor defined in the type-driven framework for compositional distributional semantics (Coecke et al., 2010). In a comprehensive experiment on ten different verbs we find no significant difference between the full tensor representation and the reduced representations. The **SKMat** and **2Mat** representations have the lowest number of parameters and offer a promising avenue of research for more complex sentence structures and sentence spaces. **KKMat** and **DMat** also had high scores on some verbs, but these representations are applicable only in spaces where a single-value output is appropriate.

In experiments where we varied the amount of training data, we found that in general more concrete verbs can learn from less data. Low concreteness verbs require particular care with dataset design, since some of the seemingly random examples can be plausible. This problem may be circumvented by using semi-supervised learning techniques.

We also found that simple numerical techniques, such as column normalisation, can markedly alter the values and quality of learning. On our data, column normalisation has a side-effect of removing the negative values that were introduced by the use of tTest weighting measure. The use of the PPMI weighting scheme and non-negative matrix factorisation (NMF) (Grefenstette et al., 2013; Van de Cruys, 2010) could lead to a similar effect, and should be investigated. Further numerical techniques for improving the estimation of the class decision boundary, and consequently the F-score, will also constitute future work.

References

Ethem Alpaydin. 1999. Combined 5x2 CV F-test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, November.

Brett W. Bader, Tamara G. Kolda, et al. 2012. Matlab tensor toolbox version 2.5. Available online, Jan.

Marco Baroni and Roberto 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)*, pages 1183–1193, Cambridge, MA.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9:5–110.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*, pages 1–8.

Nathanael Chambers and Dan Jurafsky. 2010. Improving the use of pseudo-words for evaluating selectional preferences. In *Proceedings of ACL 2010*, Uppsala, Sweden.

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

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Stephen Clark. 2013. Type-driven syntax and semantics for composing meaning vectors. In Chris Heunen, Mehrnoosh Sadrzadeh, and Edward Grefenstette, editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*, pages 359–377. Oxford University Press.

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

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen 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.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Dordrecht.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics*, pages 241–247, Atlanta,Georgia.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland, UK, July.

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).*

Edward Grefenstette. 2013. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. Ph.D. thesis, University of Oxford.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Jayant Krishnamurthy and Tom M Mitchell. 2013. Vector space semantic parsing: A framework for compositional vector space models. In *Proceedings of the 2013 ACL Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML).*

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A type-driven tensor-based semantics for CCG. In *Proceedings of the EACL 2014 Type Theory and Natural Language Semantics Workshop (TTNLS)*, Gothenburg, Sweden.

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. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08*, pages 236–244, Columbus, OH.

Tamara Polajnar and Stephen Clark. 2014. Improving distributional semantic vectors through context selection and normalisation. In *14th Conference of the European Chapter of the Association for Computational Linguistics, EACL'14*, Gothenburg, Sweden.

M. Ranzato, A. Krizhevsky, and G. E. Hinton. 2010. Factored 3-way restricted boltzmann machines for modeling natural images. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Diarmuid O Seaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of ACL 2010*, Uppsala, Sweden.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1201–1211, Jeju, Korea.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng, and Chris Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, Seattle, USA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2009)*, Vancouver, Canada.

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

Aydın Ulaş, Olcay Taner Yıldız, and Ethem Alpaydın. 2012. Cost-conscious comparison of supervised learning algorithms over multiple data sets. *Pattern Recognition*, 45(4):1772–1781, April.

Tim Van de Cruys, Laura Rimell, Thierry Poibeau, and Anna Korhonen. 2012. Multi-way tensor factorization for unsupervised lexical acquisition. In *Proceedings of COLING 2012*, Mumbai, India.

Tim Van de Cruys. 2010. A non-negative tensor factorization model for selectional preference induction. *Journal of Natural Language Engineering*, 16(4):417–437.