A Generative Model of Vector Space Semantics

Jacob Andreas
Computer Laboratory
University of Cambridge
jda33@cam.ac.uk

Zoubin Ghahramani
Department of Engineering
University of Cambridge
zoubin@eng.cam.ac.uk

Abstract

We present a novel compositional, generative model for vector space representations of meaning. This model reformulates earlier tensor-based approaches to vector space semantics as a top-down process, and provides efficient algorithms for transformation from natural language to vectors and from vectors to natural language. We describe procedures for estimating the parameters of the model from positive examples of similar phrases, and from distributional representations, then use these procedures to obtain similarity judgments for a set of adjective-noun pairs. The model's estimation of the similarity of these pairs correlates well with human annotations, demonstrating a substantial improvement over several existing compositional approaches in both settings.

1 Introduction

Vector-based word representations have gained enormous popularity in recent years as a basic tool for natural language processing. Various models of linguistic phenomena benefit from the ability to represent words as vectors, and vector space word representations allow many problems in NLP to be reformulated as standard machine learning tasks (Blei et al., 2003; Deerwester et al., 1990).

Most research to date has focused on only one means of obtaining vectorial representations of words: namely, by representing them *distributionally*. The meaning of a word is assumed to be fully specified by "the company it keeps" (Firth, 1957), and word co-occurrence (or occasionally term-document) matrices are taken to encode this context adequately. Distributional representations have been shown to work well for a variety of different tasks (Schütze and Pedersen, 1993; Baker and McCallum, 1998).

The problem becomes more complicated when we attempt represent larger linguistic structures—multiword constituents or entire sentences—within the same vector space model. The most basic issue is one of sparsity: the larger a phrase, the less frequently we expect it to occur in a corpus, and the less data we will have from which to estimate a distributional representation. To resolve this problem, recent work has focused on *compositional* vector space models of semantics. Based on the Fregean observation that the meaning of a sentence is composed from the individual meanings of its parts (Frege, 1892), research in compositional distributional semantics focuses on describing procedures for combining vectors for individual words in order to obtain an appropriate representation of larger syntactic constituents.

But various aspects of this account remain unsatisfying. We have a continuous semantic space in which finitely many vectors are associated with words, but no way (other than crude approximations like nearest-neighbor) to interpret the "meaning" of all the other points in the space. More generally, it's not clear that it even makes sense to talk about the meaning of sentences or large phrases in distributional terms, when there is no natural context to represent.

We can begin to address these concerns by turning the conventional account of composition in vector space semantics on its head, and describing a model for generating language from vectors in semantic space. Our approach is still compositional, in the sense that a sentence's meaning can be inferred from the meanings of its parts, but we relax the requirement that lexical items correspond to single vectors by allowing any vector. In the process, we acquire algorithms for both meaning inference and natural language generation.

Our contributions in this paper are as follows:

- A new generative, compositional model of

phrase meaning in vector space.

- A convex optimization procedure for mapping words onto their vector representations.
- A training algorithm which requires only positive examples of phrases with the same meaning.
- Another training algorithm which requires only distributional representations of phrases.
- A set of preliminary experimental results indicating that the model performs well on real-world data in both training settings.

2 Model overview

2.1 Motivations

The most basic requirement for a vector space model of meaning is that whatever distance metric it is equipped with accurately model human judgments of semantic similarity. That is: sequences of words which are judged to "mean the same thing" should cluster close together in the semantic space, and totally unrelated sequences of words should be spread far apart.

Beyond this, of course, inference of vector space representations should be tractable: we require efficient algorithms for analyzing natural language strings into their corresponding vectors, and for estimating the parameters of the model that does the mapping. For some tasks, it is also useful to have an algorithm for the opposite problem—given a vector in the semantic space, it should be possible to produce a natural-language string encoding the meaning of that vector; and, in keeping with our earlier requirements, if we choose a vector close to the vector corresponding to a known string, the resulting interpretation should be judged by a human to mean the same thing, and perhaps with some probability be exactly the same.

It is these three requirements—the use of human similarity judgments as the measure of the semantic space's quality, and the existence of efficient algorithms for both generation and inference—that motivate the remainder of this work.

We take as our starting point the general program of Coecke et al. (2010) which suggests that the task of analyzing into a vector space should be driven by syntax. In this framework, the compositional process consists of repeatedly combining vector space word representations according to linguistic rules, in a *bottom-up* process for translating a natural language string to a vector in space.

But our requirement that all vectors be translatable into meanings—that we have both analysis and generation algorithms—suggests that we should take the opposite approach, working with a *top down* model of vector space semantics.

For simplicity, our initial presentation of this model, and the accompanying experiments, will be restricted to the case of adjective-noun pairs. Section 5 will then describe how this framework can be extended to full sentences.

2.2 Preliminaries

We want to specify a procedure for mapping a natural language noun-adjective pair (a, n) into a vector space which we will take to be $\mathbb{R}^p$. We assume that our input sentence has already been assigned a single CCG parse (Steedman and Baldridge, 2011), which for noun-adjective pairs has the form

$$\dfrac{\underline{\textit{blue}}\ \ \underline{\textit{orangutans}}}{\dfrac{\text{N/N}\qquad\text{N}}{\text{N}}>} \tag{1}$$

Here, the parser has assigned each token a *category* of the form N, N/N, etc. Categories are either *simple*, drawn from a set of base types (here just N for "noun"), or *complex*, formed by combining simple categories. A category of the form X/Y "looks right" for a category of the form Y, and can combine with other constituents by application (we write $\text{X/Y}\ \ \text{Y} \Rightarrow \text{X}$) or composition ($\text{X/Y}\ \ \text{Y/Z} \Rightarrow \text{X/Z}$) to form higher-level constituents.

To this model we add a vector space semantics. We begin with a brief review the work of Coecke et al. (2010). Having assigned simple categories to vector spaces (in this case, N to $\mathbb{R}^p$), complex categories correspond to spaces of *tensors*. A category of the form X/Y is recursively associated with $\mathbb{S}_\text{X} \otimes \mathbb{S}_\text{Y}$, where $\mathbb{S}_\text{X}$ and $\mathbb{S}_\text{Y}$ are the tensor spaces associated with the categories X and Y respectively. So the space of adjectives (of type N/N) is just $\mathbb{R}^q \otimes \mathbb{R}^q$, understood as the set of $q \times q$ matrices. To find the meaning of a adjective-noun pair, we simply multiply the adjective matrix and noun vector as specified by the CCG derivation. The result is another vector in the same semantic space as the noun, as desired.

To turn this into a top-down process, we need to describe a procedure for splitting meanings and their associated categories.

2.3 Generation

Our goal in this subsection is to describe a probabilistic generative process by which a vector in a semantic space is realized in natural language.

Given a constituent of category X, and a corresponding vector x residing in some $\mathbb{S}_X$, we can either generate a lexical item of the appropriate type or probabilistically draw a CCG derivation rooted in X, then independently generate the leaves. For noun-adjective pairs, this can only be done in one way, namely as in (1) (for a detailed account of generative models for CCG see Hockenmaier and Steedman (2002)). We will assume that this CCG derivation tree is observed, and concern ourselves with filling in the appropriate vectors and lexical items. This is a strong independence assumption! It effectively says "the grammatical realization of a concept is independent of its meaning". We will return to it in Section 6.

The adjective-noun model has four groups of parameters: (1) a collection $\Theta_{\text{N/N}}$ of weight vectors θ_a for adjectives a, (2) a collection Θ_{N} of weight vectors θ_n for nouns n, (3) a collection $\mathcal{E}_{\text{N/N}}$ of adjective matrices E_a for adjectives a, and finally (4) a noise parameter σ^2. For compactness of notation we will denote this complete set of parameters $\mathbf{\Theta}$.

Now we can describe how to generate an adjective-noun pair from a vector x. The CCG derivation tells us to produce a noun and an adjective, and the type information further informs us that the adjective acts as a functor (here a matrix) and the noun as an argument. We begin by choosing an adjective a conditional on x. Having made our lexical choice, we deterministically select the corresponding matrix E_a from $\mathcal{E}_{\text{N/N}}$. Next we noisily generate a new vector $y = E_a x + \varepsilon$, a vector in the same space as x, corresponding to the meaning of x *without* the semantic content of a. Finally, we select a noun n conditional on y, and output the noun-adjective pair (a, n). To use the previous example, suppose x means *blue orangutans*. First we choose an adjective $a =$ "blue" (or with some probability "azure" or "ultramarine"), and select a corresponding adjective E_a. Then the vector $y = E_a x$ should mean *orangutan*, and when we generate a noun conditional on y we should have $n =$ "orangutan" (or perhaps "monkey", "primate", etc.).

This process can be summarized with the graphical model in Figure 1. In particular, we draw a

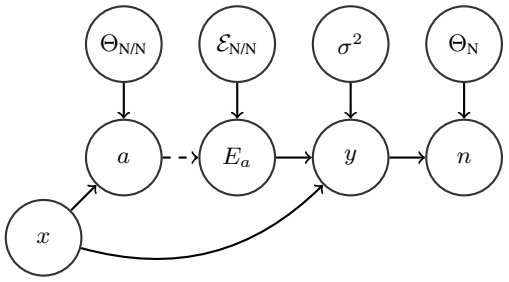


Figure 1: Graphical model of the generative process.

from a log-linear distribution over all words of the appropriate category, and use the corresponding E_a with Gaussian noise to map x onto y:

$$p(a|x; \Theta_{\text{N/N}}) = \frac{\exp(\theta_a^\top x)}{\sum_{\theta' \in \Theta_{\text{N/N}}} \exp(\theta'^\top x)} \quad (2)$$

$$p(y|x, E_a; \sigma^2) = \mathcal{N}(E_a x, \sigma^2)(y) \quad (3)$$

Last we choose n as

$$p(n|y; \Theta_{\text{N}}) = \frac{\exp(\theta_n^\top y)}{\sum_{\theta' \in \Theta_{\text{N}}} \exp(\theta'^\top z)} \quad (4)$$

Some high-level intuition about this model: in the bottom-up account, operators (drawn from tensor spaces associated with complex categories) can be thought of as acting on simple objects and "adding" information to them. (Suppose, for example, that the dimensions of the vector space correspond to actual perceptual dimensions; in the bottom-up account the matrix corresponding to the adjective "red" should increase the component of an input vector that lies in dimension corresponding to redness.) In our account, by contrast, matrices *remove* information, and the "red" matrix should act to reduce the vector component corresponding to redness.

2.4 Analysis

Now we must solve the opposite problem: given an input pair (a, n), we wish to map it to an appropriate vector in $\mathbb{R}^p$. We assume, as before, that we already have a CCG parse of the input. Then, analysis corresponds to solving the following optimization problem:

$$\arg\min_x \quad -\log p(x|a, n; \mathbf{\Theta})$$

By Bayes' rule,

$$p(x|a, n; \mathbf{\Theta}) \propto p(a, n|x; \mathbf{\Theta})p(x)$$

so it suffices to minimize

$$-\log p(x) - \log p(a, n|x; \mathbf{\Theta})$$

To find the single best complete derivation of an input pair (equivalent to the Viterbi parse tree in syntactic parsing), we can rewrite this as

$$\arg\min_{x,y} \quad -\log p(x) - \log p(a, b, y|x; \mathbf{\Theta}) \quad (5)$$

where, as before, y corresponds to the vector space semantics representation of the noun alone. We take our prior $\log p(x)$ to be a standard normal. We have:

$$\begin{aligned}
&-\log p(a, n, y|x) \\
&\quad = -\log p(a|x; \mathbf{\Theta}) - \log p(y|a, x; \mathbf{\Theta}) \\
&\quad\quad - \log p(n|y; \mathbf{\Theta}) \\
&\quad \propto -\theta_a^\top x + \log \sum_{\theta' \in \Theta_{N/N}} \exp \theta'^\top x \\
&\quad\quad + \frac{1}{\sigma^2}||E_a x - y||^2 \\
&\quad\quad - \theta_n^\top y + \log \sum_{\theta' \in \Theta_N} \exp \theta'^\top y
\end{aligned}$$

Observe that this probability is convex: it consists of a sum of linear terms, Euclidean norms, and log-normalizers, all convex functions. Consequently, Equation 5 can be solved exactly and efficiently using standard convex optimization tools (Boyd and Vandenberghe, 2004).

3 Relation to existing work

The approach perhaps most closely related to the present work is the bottom-up account given by Coecke et al. (2010), which has already been discussed in some detail in the preceding section. A regression-based training procedure for a similar model is given by Grefenstette et al. (2013). Other work which takes as its starting point the decision to endow some (or all) lexical items with matrix-like operator semantics include that of Socher et al. (2012) and Baroni and Zamparelli (2010). Indeed, it is possible to think of the model in Baroni and Zamparelli's paper as corresponding to a training procedure for a special case of this model, in which the positions of both nouns and noun-adjective vectors are fixed in advance, and in which no lexical generation step takes place. The adjective matrices learned in that paper correspond to the inverses of the E matrices used above.

Also relevant here is the work of Mitchell and Lapata (2008) and Zanzotto et al. (2010), which provide several alternative procedures for composing distributional representations of words, and Wu et al. (2011), which describes a compositional vector space semantics with an integrated syntactic model. Our work differs from these approaches in requiring only positive examples for training, and in providing a mechanism for generation as well as parsing. Other *generative* work on vector space semantics includes that of Hermann et al. (2012), which models the distribution of noun-noun compounds. This work differs from the model that paper in attempting to generate complete natural language strings, rather than simply recover distributional representations.

In training settings where we allow all positional vectors to be free parameters, it's possible to view this work as a kind of linear relational embedding (Paccanaro and Hinton, 2002). It differs from that work, obviously, in that we are interested in modeling natural language syntax and semantics rather than arbitrary hierarchical models, and provide a mechanism for realization of the embedded structures as natural language sentences.

4 Experiments

Since our goal is to ensure that the distance between natural language expressions in the vector space correlates with human judgments of their relatedness, it makes sense to validate this model by measuring precisely that correlation. In the remainder of this section, we provide evidence of the usefulness of our approach by focusing on measurements of the similarity of adjective-noun pairs (ANs). We describe two different parameter estimation procedures for different kinds of training data.

4.1 Learning from matching pairs

We begin by training the model on *matching pairs*. In this setting, we start with a collection N sets of up to M adjective-noun pairs $(a_{i1}, n_{i1}), (a_{i2}, n_{i2}), \ldots$ which mean the same thing. We fix the vector space representation y_i of each noun n_i distributionally, as described below, and find optimal settings for the lexical choice parameters $\Theta_{\text{N/N}}$ and Θ_{N}, matrices (here all $q \times q$)

$\mathcal{E}_{\text{N/N}}$, and, for each group of adjective-noun pairs in the training set, a latent representation x_i. The fact that the vectors y_i are tied to their distributional vectors does not mean we have committed to the distributional representation of the corresponding nouns! The final model represents lexical choice only with the weight vectors Θ—fixing the vectors just reduces the dimensionality of the parameter estimation problem and helps steer the training algorithm toward a good solution. The noise parameter then acts as a kind of slack variable, modeling the fact that there may be no parameter setting which reproduces these fixed distributional representations through exact linear operations alone.

We find a maximum-likelihood estimate for these parameters by minimizing

$$\mathcal{L}(\mathbf{\Theta}, x) = -\sum_{i=1}^{N}\sum_{i=1}^{M} \log p(a_{ij}, n_{ij}|x_i; \mathbf{\Theta}) \quad (6)$$

The latent vectors x_i are initialized to one of their corresponding nouns, adjective matrices E are initialized to the identity. The components of Θ_{N} are initialized identically to the nouns they select, and the components of $\Theta_{\text{N/N}}$ initialized randomly. We additionally place an L_2 regularization penalty on the scoring vectors in both Θ (to prevent weights from going to infinity) and $\mathcal{E}$ (to encourage adjectives to behave roughly like the identity). These penalties, as well as the noise parameter, are initially set to 0.1.

Note that the training objective, unlike the analysis objective, is non-convex. We use L-BFGS (Liu and Nocedal, 1989) on the likelihood function described above with ten such random restarts, and choose the parameter setting which assigns the best score to a held-out cross-validation set. Computation of the objective and its gradient at each step is linear in the number of training examples and quadratic in the dimensionality of the vector space.

Final evaluation is performed by taking a set of pairs of ANs which have been assigned a similarity score from 1–6 by human annotators. For each pair, we map it into the vector space as described in Section 2.4 above. and finally compute the cosine similarity of the two pair vectors. Performance is measured in the correlation (Spearman's ρ) between these cosine similarity scores and the human similarity judgments.

4.1.1 Setup details

Noun vectors y_i are estimated distributionally from a corpus of approximately 10 million tokens of English-language Wikipedia data (Wikimedia Foundation, 2013). A training set of adjective-noun pairs are collected automatically from a collection of reference translations originally prepared for a machine translation task. For each foreign sentence we have four reference translations produced by different translators. We assign POS tags to each reference (Loper and Bird, 2002) then add to the training data any adjective that appears exactly once in multiple reference translations, with all the nouns that follow it (e.g. "great success", "great victory", "great accomplishment"). We then do the same for repeated nouns and the adjectives that precede them (e.g. "great success", "huge success", "tremendous success"). This approach is crude, and the data collected are noisy, featuring such "synonym pairs" as ("incomplete myths", "incomplete autumns") and ("similar training", "province-level training"), as well as occasional pairs which are not adjective-noun pairs at all (e.g. "first parliamentary"). Nevertheless, as results below suggest, they appear to be good enough for purposes of learning an appropriate representation.

For the experiments described in this section, we use 500 sets of such adjective-noun pairs, corresponding to 1104 total training examples. Testing data consists of the subset of entries in the dataset from (Mitchell and Lapata, 2010) for which both the adjective and noun appear at least once (not necessarily together) in the training set, a total of 396 pairs. None of the pairs in this test set appears in training. We additionally withhold from this set the ten pairs assigned a score of 6 (indicating exact similarity), setting these aside for cross-validation.

In addition to the model discussed in the first section of this paper (referred to here as "GEN"), we consider a model in which there is only one adjective matrix E used regardless of the lexical item (referred to as "GEN-1").

The NP space is taken to be $\mathbb{R}^{20}$, and we reduce distributional vectors to 20 dimensions using a singular value decomposition.

4.2 Learning from distributional representations

While the model does not require distributional representations of latent vectors, it's useful to consider whether it can also provide a generative analog to recent models aimed explicitly at producing vectorial representations of phrases given only distributional representations of their constituent words. To do this, we take as our training data a set of N single ANs, paired with a distributional representation of each AN. In the new model, the meaning vectors x are no longer free parameters, but fully determined by these distributional representations. We must still obtain estimates for each Θ and $\mathcal{E}_{\text{N/N}}$, which we do by minimizing

$$\mathcal{L}(\mathbf{\Theta}) = -\sum_{i=1}^{N} \log p(a_{i,j}, n_{i,j} | x_i; \mathbf{\Theta}) \quad (7)$$

4.2.1 Experimental setup

Experimental setup is similar to the previous section; however, instead of same-meaning pairs collected from a reference corpus, our training data is a set of distributional vectors. We use the same noun vectors, and obtain these new latent pair vectors by estimating them in the same fashion from the same corpus.

In order to facilitate comparison with the other experiment, we collect all pairs (a_i, n_i) such that both a_i and n_i appear in the training set used in Section 4.1 (although, once again, not necessarily together). Initialization of Θ and $\mathcal{E}$, regularization and noise parameters, as well as the cross-validation procedure, all proceed as in the previous section. We also use the same restricted evaluation set, again to allow the results of the two experiments to be compared. We evaluate by measuring the correlation of cosine similarities in the learned model with human similarity judgments, and as before consider a variant of the model in which a single adjective matrix is shared.

4.3 Results

Experimental results are displayed in Table 1. For comparison, we also provide results for a baseline which uses a distributional representation of the noun only, the Adjective-Specific Linear Map (ALM) model of Baroni and Zamparelli (2010) and two vector-based compositional models discussed in (Mitchell and Lapata, 2008): $\odot$, which takes the Hadamard (elementwise) product of the distributional representations of the adjective and noun, and $+$, which adds the distributions. As before, we use SVD to project these distributional representations onto a 20-dimensional subspace.

We observe that in both matrix-based learning settings, the GEN model or its parameter-tied variant achieves the highest score (though the distributionally-trained GEN-1 doesn't perform as well as the summing approach). The pair-trained model performs best overall. All correlations except $\odot$ and the distributionally-trained GEN are statistically significant ($p < 0.05$), as are the differences in correlation between the matching-pairs-trained GEN and all other models, and between the distributionally-trained GEN-1 and ALM. Readers familiar with other papers employing the similarity-judgment evaluation will note that scores here are uniformly lower than reported elsewhere; we attribute this to the comparatively small training set (with hundreds, instead of thousands or tens of thousands of examples). This is particularly notable in the case of the ALM model, which Baroni and Zamparelli report outperforms the noun baseline when given a training set of sufficient size.

Training data	Model	ρ
Word distributions	*Noun*	.185
	$+$	.239
	$\odot$	.000
Matching pairs	GEN-1	.130
	GEN	**.365**
Word and phrase distributions	ALM	.136
	GEN-1	.201
	GEN	.097

Table 1: Results for the similarity judgment experiment.

We also give a brief demonstration of the generation capability of this model as shown in Figure 2. We demonstrate generation from three different vectors: one inferred as the latent representation of "basic principles" during training, one obtained by computing a vectorial representation of "economic development" as described in Section 2.4 and one selected randomly from within vector space. We observe that the model correctly identifies the adjectives "fundamental" and "main" as synonymous with "basic" (at least when applied to "principles"). It is also able to correctly map the vector associated with "economic

Input	Realization
Training vector ("basic principles")	tyrannical principles fundamental principles main principles
Test vector ("economic development")	economic development economic development economic development
Random vector	vital turning further obligations bad negotiations

Figure 2: Generation examples using the GEN model trained with matching pairs.

development" back onto the correct lexical realization. Words generated from the random vector appear completely unrelated; this suggests that we are sampling a portion of the space which does not correspond to any well-defined concept.

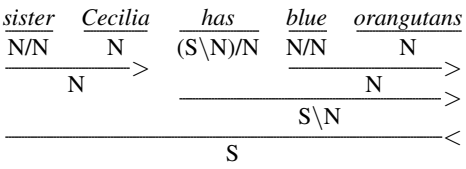
4.4 Discussion

These experimental results demonstrate, first and foremost, the usefulness of a model that is not tied to distributional representations of meaning vectors: as the comparatively poor performance of the distribution-trained models shows, with only a small number of training examples it is better to let the model invent its own latent representations of the adjective-noun pairs.

It is somewhat surprising, in the experiments with distributional training data, that the single-adjective model outperforms the multiple-adjective model by so much. We hypothesize that this is due to a search error—the significantly expanded parameter space of the multiple-adjective model makes it considerably harder to estimate parameters; in the case of the distribution-only model it is evidently so hard the model is unable to identify an adequate solution even over multiple training runs.

5 Extending the model

Having described and demonstrated the usefulness of this model for capturing noun-adjective similarity, we now describe how to extend it to capture arbitrary syntax. While appropriate experimental evaluation is reserved for future work, we outline the formal properties of the model here. We'll take as our example the following CCG derivation:



Observe that "blue orangutans" is generated according to the noun-adjective model already described.

5.1 Generation

To handle general syntax, we must first extend the set $\mathcal{E}_{\text{N/N}}$ of adjective matrices to sets $\mathcal{E}_{\text{X}}$ for all functor categories X, and create an additional set of weight vectors Θ_{X} for every category X.

When describing how to generate one split in the CCG derivation (e.g. a constituent of type S into constituents of type NP and S\NP), we can identify three cases. The first, "fully-lexicalized" case is the one already described, and is the generative process by which the a vector meaning *blue orangutans* is transformed into "blue" and "orangutans", or *sister Cecilia* into "sister" and "Cecilia". But how do we get from the top-level sentence meaning to a pair of vectors meaning *sister Cecilia* and *has blue orangutans* (an "unlexicalized" split), and from *has blue orangutans* to the word "has" and a vector meaning *blue orangutans* (a "half-lexicalized" split)?

Unlexicalized split We have a vector x with category X, from which we wish to obtain a vector y with category Y, and z with category Z. For this we further augment the sets $\mathcal{E}$ with matrices indexed by category rather than lexical item. Then we produce $y = E_{\text{Y}}x + \varepsilon$, $z = E_{\text{Z}} + \varepsilon$ where, as in the previous case, ε is Gaussian noise with variance σ^2. We then recursively generate subtrees from y and z.

Half-lexicalized split This proceeds much as in the fully lexicalized case. We have a vector x from which we wish to obtain a vector y with category Y, and a lexical item w with category Z.

We choose w according to Equation 2, select a matrix E_w and produce $y = E_w x + \varepsilon$ as before, and then recursively generate a subtree from y without immediately generating another lexical item for y.

5.2 Analysis

As before, it suffices to minimize $-\log p(x) - \log p(W, P|x)$ for a sentence

$W = (w_1, w_2, \cdots, w_n)$ and a set of internal vectors P. We select our prior $p(x)$ exactly as before, and can define $p(W, P|x)$ recursively. The fully-lexicalized case is exactly as above. For the remaining cases, we have:

Unlexicalized split Given a subsequence $W_{i:j} = (w_i, \cdots, w_j)$, if the CCG parse splits $W_{i:j}$ into constituents $W_{i:k}$ and $W_{k:j}$, with categories Y and Z, we have:

$$\begin{aligned} -\log p(W_{i:j}|x) = &-\log p(W_{i:k}, P|E_Y x) \\ &-\log p(W_{k:j}, P|E_Z x) \end{aligned}$$

Half-lexicalized split If the parse splits $W_{i:j}$ into w_i and $W_{i+1:j}$ with categories Y and Z, and $y \in P$ is the intermediate vector used at this step of the derivation, we have:

$$\begin{aligned} &-\log p(W_{i:j}, y|x) \\ &\quad = -\log p(w_i|x) - \log p(y|x, w_i) \\ &\quad\quad -\log p(W_{i+1:j}|y) \\ &\quad \propto -\theta_{w_i}^T x + \log \sum_{w' \in L_Y} \exp \theta_{w'}^T x \\ &\quad\quad + \frac{1}{\sigma^2} ||E_{w_i} x - y||^2 \\ &\quad\quad -\log p(W_{i+1:j}, P|y) \end{aligned}$$

Finally, observe that the complete expression of the log probability of any derivation is, as before, a sum of linear and convex terms, so the optimization problem remains convex for general parse trees.

6 Future work

Various extensions to the model proposed in this paper are possible. The fact that relaxing the distributional requirement for phrases led to performance gains suggests that something similar might be gained from nouns. If a reliable training procedure could be devised with noun vectors as free parameters, it might learn an even better model of phrase similarity—and, in the process, simultaneously perform unsupervised word sense disambiguation on the training corpus.

Unlike the work of Coecke et al. (2010), the structure of the types appearing in the CCG derivations used here are neither necessary nor sufficient to specify the form of the matrices used in this paper. Instead, the function of the CCG derivation is simply to determine which words should be assigned matrices, and which nouns. While CCG provides a very natural way to do this, it is by no means the only way, and future work might focus on providing an analog using a different grammar—all we need is a binary-branching grammar with a natural functor-argument distinction.

Finally, as mentioned in Section 2.3, we have made a significant independence assumption in requiring that the entire CCG derivation be generated in advance. This assumption was necessary to ensure that the probability of a vector in meaning space given its natural language representation would be a convex program. We suspect, however, that it is possible to express a similar probability for an entire packed forest of derivations, and optimize it globally by means of a CKY-like dynamic programming approach. This would make it possible to optimize simultaneously over all possible derivations of a sentence, and allow positions in meaning space to influence the form of those derivations.

7 Conclusion

We have introduced a new model for vector space representations of word and phrase meaning, by providing an explicit probabilistic process by which natural language expressions are generated from vectors in a continuous space of meanings. We've given efficient algorithms for both analysis into and generation out of this meaning space, and described two different training procedures for estimating the parameters of the model. Experimental results demonstrate that these algorithms are capable of modeling graded human judgments of phrase similarity given only positive examples of matching pairs, or distributional representations of pairs as training data; when trained in this fashion, the model outperforms several other compositional approaches to vector space semantics. We have concluded by suggesting how syntactic information might be more closely integrated into this model. While the results presented here are preliminary, we believe they present compelling evidence of representational power, and motivate further study of related models for this problem.

Acknowledgments

We would like to thank Stephen Clark and Andreas Vlachos for feedback on a draft of this paper.

References

L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

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

John Rupert Firth. 1957. *A synopsis of linguistic theory, 1930-1955*.

Gottlob Frege. 1892. Uber Sinn und Bedeutung. *Zeitschrift fur Philosophie und philosophische Kritik*, pages 25–50. English Translation: em On Sense and Meaning, in Brian McGuinness (ed), em Frege: collected works, pp. 157–177, Basil Blackwell, Oxford.

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)*.

Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2012. An unsupervised ranking model for noun-noun compositionality. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 132–141. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 335–342. Association for Computational Linguistics.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

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

Alberto Paccanaro and Jefferey Hinton. 2002. Learning hierarchical structures with linear relational embedding. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, volume 14, page 857. MIT Press.

Hinrich Schütze and Jan Pedersen. 1993. A vector model for syntagmatic and paradigmatic relatedness. *Making sense of words*, pages 104–113.

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

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax Oxford: Blackwell*, pages 181–224.

Wikimedia Foundation. 2013. Wikipedia. http://dumps.wikimedia.org/enwiki/. Accessed: 2013-04-20.

Stephen Wu, William Schuler, et al. 2011. Structured composition of semantic vectors. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, pages 295–304. Citeseer.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics.