Learning Task-specific Bilexical Embeddings

Pranava Swaroop Madhyastha **Xavier Carreras** **Ariadna Quattoni**
TALP Research Center
Universitat Politècnica de Catalunya
Campus Nord UPC, Barcelona
pranava,carreras,aquattoni@lsi.upc.edu

Abstract

We present a method that learns bilexical operators over distributional representations of words and leverages supervised data for a linguistic relation. The learning algorithm exploits low-rank bilinear forms and induces low-dimensional embeddings of the lexical space tailored for the target linguistic relation. An advantage of imposing low-rank constraints is that prediction is expressed as the inner-product between low-dimensional embeddings, which can have great computational benefits. In experiments with multiple linguistic bilexical relations we show that our method effectively learns using embeddings of a few dimensions.

1 Introduction

We address the task of learning functions that compute compatibility scores between pairs of lexical items under some linguistic relation. We refer to these functions as bilexical operators. As an instance of this problem, consider learning a model that predicts the probability that an adjective modifies a noun in a sentence. In this case, we would like the bilexical operator to capture the fact that some adjectives are more compatible with some nouns than others. For example, a bilexical operator should predict that the adjective *electronic* has high probability of modifying the noun *device* but little probability of modifying the noun *case*.

Bilexical operators can be useful for multiple NLP applications. For example, they can be used to reduce ambiguity in a parsing task. Consider the following sentence extracted from a weblog: *Vynil can be applied to electronic devices and cases, wooden doors and furniture and walls*. If we want to predict the dependency structure of this sentence we need to make several decisions. In particular, the parser would need to decide (1) Does *electronic* modify *devices*? (2) Does *electronic* modify *cases*? (3) Does *wooden* modify *doors*? (4) Does *wooden* modify *furniture*? Now imagine that in the corpus used to train the parser none of these nouns have been observed, then it is unlikely that these attachments can be resolved correctly. However, if an accurate noun-adjective bilexical operator were available most of the uncertainty could be resolved. This is because a good bilinear operator would give high probability to the pairs *electronic-device*, *wooden-door*, *wooden-furniture* and low probability to the pair *electronic-case*.

The simplest way of inducing a bilexical operator is to learn it from a training corpus. That is, assuming that we are given some data annotated with a linguistic relation between a modifier and a head (e.g. adjective and noun) we can simply build a maximum likelihood estimator for $\Pr(m \mid h)$ by counting the occurrences of modifiers and heads under the target relation. For example, we could consider learning bilexical operators from sentences annotated with dependency structures. Clearly, this model can not generalize to head words not present in the training data.

To mitigate this we could consider bilexical operators that can exploit lexical embeddings, such as a distributional vector-space representation of words. In this case, we assume that for every word we can compute an n-dimensional vector space representation $\phi(w) \rightarrow \mathbb{R}^n$. This representation typically captures distributional features of the context in which the lexical item can occur. The key point is that we do not need a supervised corpus to compute the representation. All we need is a large textual corpus to compute the relevant statistics. Once we have the representation we can exploit operations in the induced vector space to define lexical compatibility operators. For example we could define a bilexical

operator as:

$$\Pr(m \mid h) = \frac{\exp\{\langle \phi(m), \phi(h) \rangle\}}{\sum_{m'} \exp\{\langle \phi(m'), \phi(h) \rangle\}} \quad (1)$$

where $\langle \phi(x), \phi(y) \rangle$ denotes the inner-product. Alternatively, given an initial high-dimensional distributional representation computed from a large textual corpus we could first induce a projection to a lower k dimensional space by performing truncated singular value decomposition. The idea is that the lower dimensional representation will be more efficient and it will better capture the relevant dimensions of the distributional representation. The bilexical operator would then take the form of:

$$\Pr(m|h) = \frac{\exp\{\langle U\phi(m), U\phi(h) \rangle\}}{\sum_{m'} \exp\{\langle U\phi(m'), U\phi(h) \rangle\}} \quad (2)$$

where $U \in \mathbb{R}^{k \times n}$ is the projection matrix obtained via SVD. The advantage of this approach is that as long as we can estimate the distribution of contexts of words we can compute the value of the bilexical operator. However, this approach has a clear limitation: to design a bilinear operator for a target linguistic relation we must design the appropriate distributional representation. Moreover, there is no clear way of exploiting a supervised training corpus.

In this paper we combine both the supervised and distributional approaches and present a learning algorithm for inducing bilexical operators from a combination of supervised and unsupervised training data. The main idea is to define bilexical operators using bilinear forms over distributional representations: $\phi(x)^\top W \phi(y)$, where $W \in \mathbb{R}^{n \times n}$ is a matrix of parameters. We can then train our model on the supervised training corpus via conditional maximum-likelihood estimation. To induce a low-dimensional representation, we first observe that the implicit dimensionality of the bilinear form is given by the rank of W. In practice controlling the rank of W can result in important computational savings in cases where one evaluates a target word x against a large number of candidate words y: this is because we can project the representations $\phi(x)$ and $\phi(y)$ down to the low-dimensional space where evaluating the function is simply an inner-product. This setting is in fact usual, for example for lexical retrieval applications (e.g. given a noun, sort all adjectives in the vocabulary according to their compatibility), or for parsing (where one typically evaluates the compatibility between all pairs of words in a sentence).

Consequently with these ideas, we propose to regularize the maximum-likelihood estimation using a nuclear norm regularizer that serves as a convex relaxation to the rank function. To minimize the regularized objective we make use of an efficient iterative proximal method that involves computing the gradient of the function and performing singular value decompositions.

We test the proposed algorithm on several linguistic relations and show that it can predict modifiers for unknown words more accurately than the unsupervised approach. Furthermore, we compare different types of regularizers for the bilexical operator W, and observe that indeed the low-rank regularizer results in the most efficient technique at prediction time.

In summary, the main contributions of this paper are:

- We propose a supervised framework for learning bilexical operators over distributional representations, based on learning bilinear forms W.
- We show that we can obtain low-dimensional compressions of the distributional representation by imposing low-rank constraints to the bilinear form. Combined with supervision, this results in lexical embeddings tailored for a specific bilexical task.
- In experiments, we show that our models generalize well to unseen word pairs, using only a few dimensions, and outperforming standard unsupervised distributional approaches. We also present an application to prepositional phrase attachment.

2 Bilinear Models for Bilexical Predictions

2.1 Definitions

Let $\mathcal{V}$ be a vocabulary, and let $x \in \mathcal{V}$ denote a word. Let $\mathcal{H} \subseteq \mathcal{V}$ be a set of head words, and $\mathcal{M} \subseteq \mathcal{V}$ be a set of modifier words. In the noun-adjective relation example, $\mathcal{H}$ is the set of nouns and $\mathcal{M}$ is the set

of adjectives.

The task is as follows. We are given a training set of l tuples $\mathcal{D} = \{(m,h)^1, \ldots, (m,h)^l\}$, where $m \in \mathcal{M}$ and $h \in \mathcal{H}$ and we want to learn a model of the conditional distribution $\Pr(m \mid h)$. We want this model to perform well on all head-modifier pairs. In particular we will test the performance of the model on heads that do not appear in $\mathcal{D}$.

We assume that we are given access to a distributional representation function $\phi : \mathcal{V} \rightarrow \mathbb{R}^n$, where $\phi(x)$ is the n-dimensional representation of x. Typically, this function is computed from an unsupervised corpus. We use $\phi(x)_{[i]}$ to refer to the i-th coordinate of the vector.

2.2 Bilinear Model

Our model makes use of the bilinear form $W : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, where $W \in \mathbb{R}^{n \times n}$, and evaluates as $\phi(m)^\top W \phi(h)$. We define the bilexical operator as:

$$\Pr(m \mid h) = \frac{\exp\left\{\phi(m)^\top W \phi(h)\right\}}{\sum_{m' \in \mathcal{M}} \exp\left\{\phi(m')^\top W \phi(h)\right\}} \tag{3}$$

Note that the above model is nothing more than a conditional log-linear model defined over n^2 features $f_{i,j}(m,h) = \phi(m)_{[i]}\phi(h)_{[j]}$ (this can be seen clearly when we write the bilinear form as $\sum_{i=1}^n \sum_{j=1}^n f_{i,j}(m,h) W_{i,j}$. The reason why it is useful to regard W as a matrix will become evident in the next section.

Before moving to the next section, let us note that the unsupervised SVD model in Eq. (2) is also a bilinear model as defined here. This can be seen if we set $W = UU^\top$, which is a bilinear form of rank k. The key difference is in the way W is learned using supervision.

3 Learning Low-rank Bilexical Operators

3.1 Low-rank Optimization

Given a training set $\mathcal{D}$ and a feature function $\phi(x)$ we can do standard conditional max-likelihood optimization and minimize the negative of the log-likelihood function, $\log \Pr(\mathcal{D})$:

$$\sum_{(m,h) \in \mathcal{D}} \phi(m)^\top W \phi(h) - \log \sum_{m' \in \mathcal{M}} \exp\left\{\phi(m')^\top W \phi(h)\right\} \tag{4}$$

We would like to control the complexity of the learned model by including some regularization penalty. Moreover, like in the low-dimensional unsupervised approach we want our model to induce a low-dimensional representation of the lexical space. The first observation is that the bilinear form computes a weighted inner product in some space. Consider the singular value decomposition: $W = U\Sigma V$. We can write the bilinear form as: $[\phi(m)^\top U]\ \Sigma\ [V\phi(h)]$, thus we can regard $\tilde{m} = \phi(m)^\top U$ as a projection of m and $\tilde{h} = V\phi(h)$ as a projection of h. Then the bilinear form can be written as: $\sum_{i=1}^n \Sigma_{[i,i]}\ \tilde{m}_{[i]}\ \tilde{h}_{[i]}$. The rank of W defines the dimensionality of the induced space. It is easy to see that if W has rank k it can be factorized as $U\Sigma V$ where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times n}$.

Since the rank of W determines the dimensionality of the induced space, it would be reasonable to add a rank minimization penalty in the objective in (4). Unfortunately this would lead to a non-convex regularized objective. Instead, we propose to use as a regularizer a convex relaxation of the rank function, the nuclear norm $\|W\|_*$ (the ℓ_1 norm of the singular values of W). Putting it all together, our learning algorithm minimizes:

$$\sum_{(m,h) \in \mathcal{D}} -\log \Pr(m \mid h)) + \lambda \|W\|_* \tag{5}$$

Here λ is a constant that controls the trade-off between fitting the data and the complexity of the model. This objective is clearly convex since both the objective and the regularizer are convex. To minimize it we use the a proximal gradient algorithm which is described next.

3.2 A Proximal Algorithm for Bilexical Operators

We now describe the learning algorithm that we use to induce the bilexical operators from training data. We are interested in minimizing the objective (5), or in fact a more general version where we can replace the regularizer $\|W\|_*$ by standard ℓ_1 or ℓ_2 penalties. For any convex regularizer $r(W)$ (namely ℓ_1, ℓ_2 or the nuclear norm) the objective in (5) is convex. Our learning algorithm is based on a simple optimization scheme known as *forward-backward splitting (FOBOS)* (Duchi and Singer, 2009).

This algorithm has convergence rates in the order of $1/\epsilon^2$, which we found sufficiently fast for our application. Many other optimization approaches are possible, for example one could express the regularizer as a convex constraint and utilize a projected gradient method which has a similar convergence rate. Proximal methods are slightly more simple to implement and we chose the proximal approach.

The FOBOS algorithm works as follows. In a series of iterations $t = 1 \dots T$ compute parameter matrices W_t as follows:

1. Compute the gradient of the negative log-likelihood, and update the parameters

$$W_{t+0.5} = W_t - \eta_t g(W_t)$$

   where $\eta_t = \frac{c}{\sqrt{t}}$ is a step size and $g(W_t)$ is the gradient of the loss at W_t.

2. Update $W_{t+0.5}$ to take into account the regularization penalty $r(W)$, by solving

$$W_{t+1} = \operatorname*{argmin}_W ||W_{t+0.5} - W||_2^2 + \eta_t \lambda r(W)$$

   For the regularizers we consider, this step is solved using the *proximal operator* associated with the regularizer. Specifically:

   - For ℓ_1 it is a simple thresholding:

$$W_{t+1}(i,j) = \text{sign}(W_{t+0.5}(i,j)) \cdot \max(W_{t+0.5}(i,j) - \eta_t \lambda, 0)$$

   - For ℓ_2 it is a simple scaling:

$$W_{t+1} = \frac{1}{1 + \eta_t \lambda} W_{t+0.5}$$

   - For nuclear-norm, perform SVD thresholding. Compute the SVD to write $W_{t+0.5} = USV^\top$ with S a diagonal matrix and U, V orthogonal matrices. Denote by σ_i the i-th element on the diagonal of S. Define a new matrix $\bar{S}$ with diagonal elements $\bar{\sigma}_i = \max(\sigma_i - \eta_t \lambda, 0)$. Then set

$$W_{t+1} = U\bar{S}V^\top$$

Optimizing a bilinear model using nuclear-norm regularization involves the extra cost of performing SVD of W at each iteration. In our experiments the dimension of W was $2,000 \times 2,000$ and computing SVD was fast, much faster than computing the gradient, which dominates the cost of the algorithm. The optimization parameters of the method are the regularization constant λ, the step size constant c and the number of iterations T. In our experiments we ran a range of λ and c values for 200 iterations, and used a validation set to pick the best configuration.

4 Related Work

Research in learning representations for natural language processing can be broadly classified into two different paradigms based on the learning setting: unsupervised representation learning and semi-supervised representation learning. Unsupervised representation learning does not require any supervised training data, while semi-supervised representation learning requires the presence of supervised training data with the potential advantage that it can adapt the representation to the task at hand.

Unsupervised approaches to learning representations mainly involve representations that are learned not for a specific task, rather a variety of tasks. These representations rely more on the property of

abstractness and generalization. Further, unsupervised approaches can be roughly categorized into (a) clustering-based approaches that make use of clusters induced using a notion of distributed similarity and are used as features (Liang, 2005; Koo et al., 2008; Ratinov and Roth, 2009; Zhao et al., 2009; Lin and Wu, 2009); (b) neural-network-based representations that focus on learning multilayer neural network in a way to extract features from the data (Morin and Bengio, 2005; Bengio and Sénécal, 2008; Mnih and Hinton, 2009); (c) pure distributional approaches that principally follow the distributional assumption that the words which share a set of contexts are similar (Sahlgren, 2006; Turney and Pantel, 2010; Dumais et al., 1988; Landauer et al., 1998; Lund et al., 1995; Väyrynen et al., 2007).

We also induce lexical embeddings, but in our case we employ supervision. That is, we follow a semi-supervised paradigm for learning representations. Semi-supervised approaches initially learn representations typically in an unsupervised setting and then induce a representation that is jointly learned for the task with a labeled corpus. A high-dimensional representation is extracted from unlabeled data, while the supervised step compresses the representation to be low-dimensional in a way that favors the the task at hand.

Collobert and Weston (2008) present a neural network language model, where given a sentence, it performs a set of language processing tasks (from part of speech tagging, chunking, extracting named entity, extracting semantic roles and decisions on the correctness of the sentence) by using the learned representations. The representation itself is extracted from unlabeled corpora, while all the other tasks are jointly trained on labeled corpus.

(Socher et al., 2011a; Socher et al., 2011b) present a model based on recursive neural networks that learns vector space representations for words, multi-word phrases and sentences. Given a sentence with its syntactic structure, their model assings vector representations to each of the lexical tokens of the sentence, and then traverses the syntactic tree bottom-up, such that at each node a vector representation of the corresponding phrase is obtained by composing the vectors associated with the children.

Bai et al. (2010) use a technique similar to ours, using bilinear forms with low-rank constraints. In their case, they explicitly look for a low-rank factorization of the matrix, making their optimization non-convex. As far as we know, ours is the first convex formulation, where we employ a relaxation of the rank (i.e. the nuclear norm) to make the objective convex. They apply the method to document ranking, and thus optimize a max-margin ranking loss. In our application to bilexical models, we perform conditional max-likelihood estimation. Hutchinson et al. (2013) propose an explicitly sparse and low-rank maximum-entropy language model. The sparse plus low rank setting is learned in such a way that the low rank component learns the regularities in the training data and the sparse component learns the exceptions like multiword expressions etc.

Chechik et al. (2010) also learned bilinear operators using max-margin techniques, with pairwise similarity as supervision, but they did not consider low-rank constraints.

One related area where bilinear operators are used to induce embeddings is distance metric learning. Weinberger and Saul (2009) used large-margin nearest neighbor methods to learn a non-sparse embedding, but these are computationally intensive and might not be suitable for large-scale tasks in NLP.

5 Experiments on Syntactic Relations

We conducted a set of experiments to test the ability of our algorithm to learn bilexical operators for several linguistic relations. As supervised training data we use the gold standard dependencies of the WSJ training section of the Penn Treebank (Marcus et al., 1993). We consider the following relations:

- Noun-Adjective: we model the distribution of adjectives given a noun; and a separate distribution of nouns given an adjective.
- Verb-Object: we model the distribution of object nouns given a verb; and a separate distribution of verbs given an object.
- Prepositions: in this case we consider bilexical operators associated with a preposition, which model the probability of a head noun or verb above the preposition *given* the noun below the preposition. We present results for prepositional relations given by “with”, “for”, “in” and “on”.

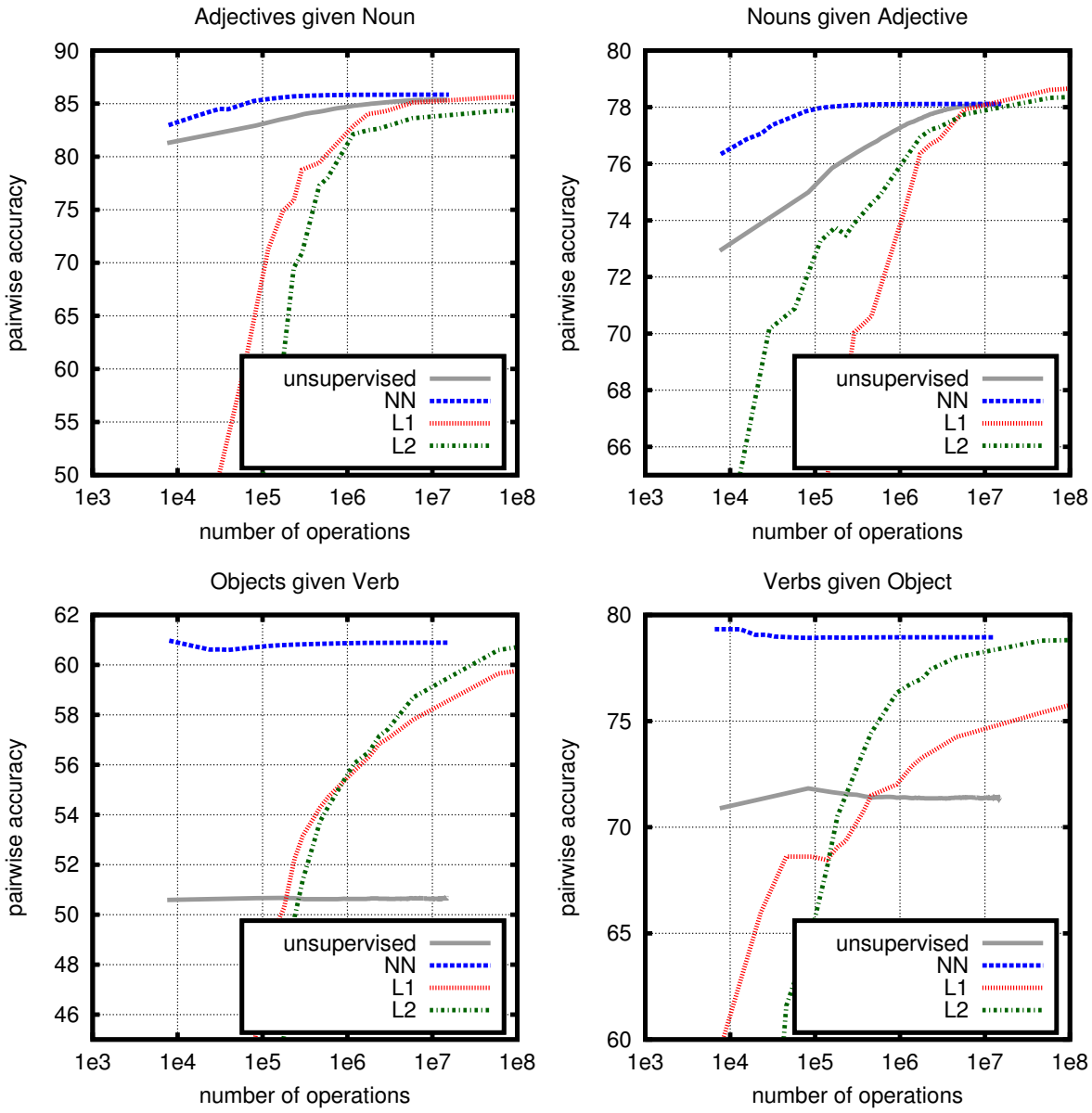

Figure 1: Pairwise accuracy with respect to the number of double operations required to compute the distribution over modifiers for a head word. Plots for noun-adjective and verb-object relations, in both directions.

The distributional representation $\phi(x)$ was computed using the BLLIP corpus (Charniak et al., 2000). We compute a bag-of-words representation for the context of each lexical item, that is $\phi(w)_{[i]}$ corresponds to the frequency of word i appearing in the context of w. We use a context window of size 10 and restrict our bag-of-words vocabulary to contain only the 2,000 most frequent words present in the corpus. Vectors were normalized.

To test the performance of our algorithm for each relation we partition the set of heads into a training and a test set, 60% of the heads are use for training, 10% of the heads are used for validation and 30% of the heads are used for testing. Then, we consider all observed modifiers in the data to form a vocabulary of modifier words. The goal of this task is to learn conditional distribution over all these modifers given a head word without context. In our experiments, the number of modifiers per relation ranges from 2,500 to 7,500 words. For each head word, we create a list of *compatible* modifiers from the annotated data, by taking all modifiers that occur at least once with the head. Hence, for each head the set of all modifiers is partitioned into compatible and non-compatible. For testing, we measure a *pairwise accuracy*, the

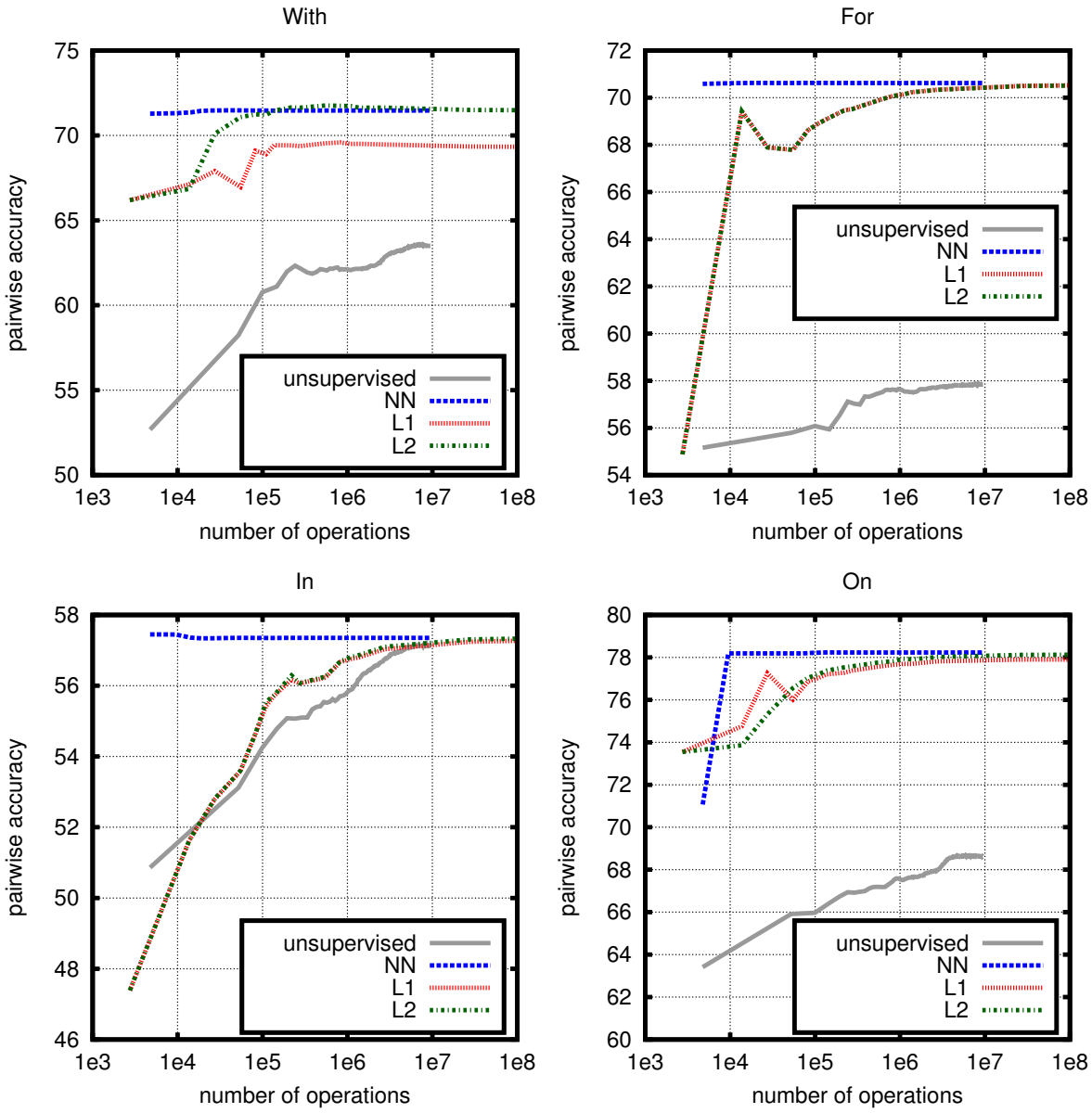

Figure 2: Pairwise accuracy with respect to the number of double operations required to compute the distribution over modifiers for a head word. Plots for four prepositional relations: with, for, in, on. The distributions are of verbs and objects above the preposition given the noun below the preposition.

percentage of compatible/non-compatible pairs of modifiers where the former obtains higher probability. Let us stress that none of the test head words has been observed in training, while the list of modifiers is the same for training, validation and testing.

We compare the performance of the bilexical model trained with nuclear norm regularization (NN) with other regularization penalties (L1 and L2). We also compare these supervised methods with an unsupervised model: a low-dimensional SVD model as in Eq. (2), which corresponds to an inner product as in Eq. (1) when all dimensions are considered.

To report performance, we measure pairwise accuracy with respect to the capacity of the model in terms of number of active parameters. To measure the capacity of a model we consider the number of double operations that are needed to compute, given a head, the scores for all modifiers in the vocabulary (we exclude the exponentiations and normalization needed to compute the distribution of modifiers given a head, since this is a constant cost for all the models we compare, and is not needed if we only want to rank modifiers). Recall that the dimension of $\phi(x)$ is n, and assume that there are m total modifiers in

Noun	Predicted Adjectives
president	executive, senior, chief, frank, former, international, marketing, assistant, annual, financial
wife	former, executive, new, financial, own, senior, old, other, deputy, major
shares	annual, due, net, convertible, average, new, high-yield, initial, tax-exempt, subordinated
mortgages	annualized, annual, three-month, one-year, average, six-month, conventional, short-term, higher, lower
month	last, next, fiscal, first, past, latest, early, previous, new, current
problem	new, good, major, tough, bad, big, first, financial, long, federal
holiday	new, major, special, fourth-quarter, joint, quarterly, third-quarter, small, strong, own

Table 1: 10 most likely adjectives for some test nouns.

the vocabulary. In our experiments $n = 2,000$ and m ranges from $2,500$ to $7,500$. The correspondances with operations are:

- Assume that the L1 and L2 models have k non-zero weights in W. Then the number of operations to compute a distribution is km.
- Assume that the NN and the unsupervised models have rank k. We assume that the modifier vectors are alredy projected down to k dimensions. For a new head, one needs to project it and perform m inner products, hence the number of operations is $kn + km$.

Figure 1 shows the performance of models for noun-adjective and verb-object relations, while Figure 2 shows plots for prepositional relations. [1] The first observation is that supervised approaches outperform the unsupervised approach. In cases such as noun-adjetive relations the unsupervised approach performs close to the supervised approaches, suggesting that the pure distributional approach can sometimes work. But in most relations the improvement obtained by using supervision is very large. When comparing the type of regularizer, we see that if the capacity of the model is unrestricted (right part of the curves), all models tend to perform similarly. However, when restricting the size, the nuclear-norm model performs much better. Roughly, 20 hidden dimensions are enough to obtain the most accurate performances (which result in $\sim 140,000$ operations for initial representaions of $2,000$ dimensions and $5,000$ modifier candidates). As an example of the type of predictions, Table 1 shows the most likely adjectives for some test nouns.

6 Experiments on PP Attachment

We now switch to a standard classification task, prepositional phrase attachment, that we frame as a bilexical prediction task. We start from the formulation of the task as a binary classification problem by Ratnaparkhi et al. (1994): given a tuple $x = \langle v, o, p, n \rangle$ consisting of a verb v, noun object o, preposition p and noun n, decide if the prepositional phrase p-n attaches to v ($y = \text{V}$) or to o ($y = \text{O}$). For example, in $\langle$ `meet,demand,for,products` $\rangle$ the correct attachment is O.

Ratnaparkhi et al. (1994) define a linear maximum likelihood model of the form $\Pr(y \mid x) = \exp\{\langle w, f(x,y)\rangle\} * Z(x)^{-1}$, where $f(x,y)$ is a vector of d features, w is a parameter vector in $\mathbb{R}^d$, and $Z(x)$ is the normalizer summing over $y = \{\text{V}, \text{O}\}$. Here we define a bilexical model of the form that uses a distributional representation ϕ:

$$\Pr(\text{V}|\langle v,o,p,n\rangle) = \frac{\exp\{\phi(v)^\top W^p_{\text{V}}\, \phi(n)\}}{Z(x)} \qquad \Pr(\text{O}|\langle v,o,p,n\rangle) = \frac{\exp\{\phi(o)^\top W^p_{\text{O}}\, \phi(n)\}}{Z(x)} \tag{6}$$

[1] To obtain curves for each model type with respect to a range of number of operations, we first obtained the best model on validation data and then forced it to have at most k non-zero features or rank k by projecting, for a range of k values.

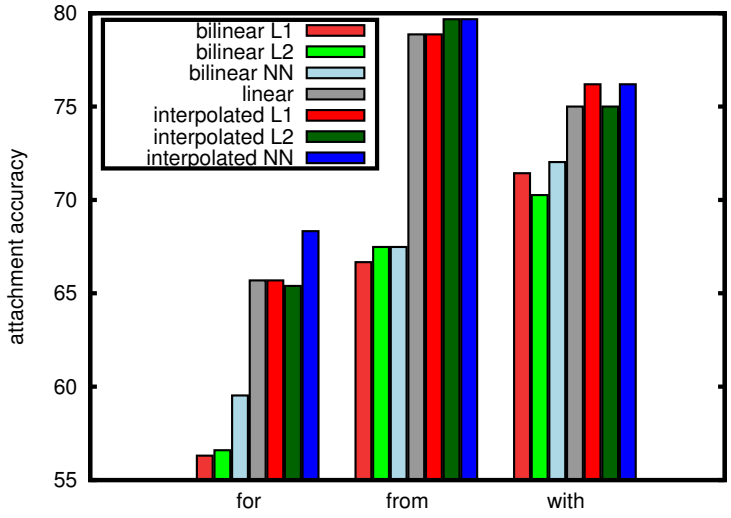

Figure 3: Attachment accuracies of linear, bilinear and interpolated models for three prepositions.

The bilinear model is parameterized by two matrices W_{V} and W_{O} per preposition, each of which captures the compatibility between nouns below a certain preposition and heads of V or O prepositional relations, respectively. Again $Z(x)$ is the normalizer summing over $y = \{\text{V}, \text{O}\}$, but now using the bilinear form. It is straighforward to modify the learning algorithm in Section 3 such that the loss is a negative log-likelihood for binary classification, and the regularizer considers the sum of norms of the model matrices.

We ran experiments using the data by Ratnaparkhi et al. (1994). We trained separate models for different prepositions, focusing on the prepositions that are more ambiguous: `for`, `from`, `with`. We compare to a linear "maxent" model following Ratnaparkhi et al. (1994) that uses the same feature set. Figure 3 shows the test results for the linear model, and bilinear models trained with L1, L2, NN regularization penalties. The results of the bilinear models are significantly below the accuracy of the linear model, suggesting that some of the non-lexical features of the linear model (such as prior weighting of the two classes) might be difficult to capture by the bilinear model over lexical representations. To check if the bilinear model might complement the linear model or just be worse than it, we tested simple combinations based on linear interpolations. For a constant $\lambda \in [0, 1]$ we define:

$$\Pr(y \mid x) = \lambda \Pr_{\text{L}}(y \mid x) + (1 - \lambda) \Pr_{\text{B}}(y \mid x) \quad . \tag{7}$$

We search for the best λ on the validation set, and report results of combining the linear model with each of the three bilinear models. Results are shown also in Figure 3. Interpolation models improve over linear models, though only the improvement for `for` is significant (2.6%). Future work should exploit finer combinations between standard linear features and distributional bilinear forms.

7 Conclusions

We have presented a model for learning bilexical operators that can leverage both supervised and unsupervised data. The model is based on exploiting bilinear forms over distributional representations. The learning algorithm induces a low-dimensional representation of the lexical space by imposing low-rank constraints on the parameters of the bilinear form. By means of supervision, our model induces two low-dimensional lexical embeddings, one on each side of the bilexical linguistic relation, and computations can be expressed as an inner-product between the two embeddings. This factorized form of the model can have great computational advantages: in many applications one needs to evaluate the function multiple times for a fixed set of lexical items, for example in dependency parsing. Hence, one can first project the lexical items to their embeddings, and then compute all pairwise scores as inner-products. In experiments, we have shown that the embeddings we obtain in a number of linguistic relations can be modeled with a few hidden dimensions.

As future work, we would like to apply the low-rank approach to other model forms that can employ lexical embeddings, specially when supervision is available. For example, dependency parsing models, or models of predicate-argument structures representing semantic roles, exploit bilexical relations. In these applications, being able to generalize to word *pairs* that are not observed during training is essential.

We would also like to study how to combine low-rank bilexical operators, which in essence induce a task-specific representation of words, with other forms of features that capture class or contextual information. One desires that such combinations can preserve the computational advantages behind low-rank embeddings.

Acknowledgements

We thank the reviewers for their helpful comments. This work was supported by projects XLike (FP7-288342), ERA-Net CHISTERA VISEN and TACARDI (TIN2012-38523-C02-00). Xavier Carreras was supported by the Ramón y Cajal program of the Spanish Government (RYC-2008-02223).

References

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. 2010. Learning to rank with (a lot of) word features. *Information Retrieval*, 13(3):291–314, June.

Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, and Mark Johnson. 2000. BLLIP 1987–89 WSJ Corpus Release 1, LDC No. LDC2000T43. Linguistic Data Consortium.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, pages 1109–1135.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 281–285. ACM.

Brian Hutchinson, Mari Ostendorf, and Maryam Fazel. 2013. Exceptions in language as learned by the multi-factor sparse plus low-rank language model. In *ICASSP*, pages 8580–8584.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038, Suntec, Singapore, August. Association for Computational Linguistics.

Kevin Lund, Curt Burgess, and Ruth A. Atchley. 1995. Semantic and associative priming in high-dimensional semantic space. In *Cognitive Science Proceedings, LEA*, pages 660–665.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS05*, pages 246–252.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, HLT '94, pages 250–255, Stroudsburg, PA, USA. Association for Computational Linguistics.

Magnus Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, January.

Jaakko J. Väyrynen, Timo Honkela, and Lasse Lindqvist. 2007. Towards explicit semantic features using independent component analysis. In *Proceedings of the Workshop Semantic Content Acquisition and Representation (SCAR)*, Stockholm, Sweden. Swedish Institute of Computer Science. SICS Technical Report T2007-06.

Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, June.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60, Boulder, Colorado, June. Association for Computational Linguistics.