Distributed Representation Prediction for Generalization to New Words

Hugo Larochelle and Yoshua Bengio
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch, Montreal, H3C 3J7, QC, Canada
{larocheh,bengioy}@iro.umontreal.ca

Technical Report 1284
Département d'Informatique et Recherche Opérationnelle

August 21st, 2006

Abstract

Learning distributed representations of symbols (e.g. words) has been used in several Natural Language Processing systems. Such representations can capture semantic or syntactic similarities between words, which permit to fight the curse of dimensionality when considering sequences of such words. Unfortunately, because these representations are learned only for a previously determined vocabulary of words, it is not clear how to obtain representations for new words. We present here an approach which gets around this problem by considering the distributed representations as predictions from low-level or domain-knowledge features of words. We report experiments on a Part Of Speech tagging task, which demonstrates the success of this approach in learning meaningful representations and in providing improved accuracy, especially for new words.

1 Introduction

A central issue in machine learning is the curse of dimensionality, which corresponds to the difficulty of training a model that can generalize well to unseen cases when the input space is of high dimensionality. Tackling this problem is especially crucial when the input space corresponds to a set of symbols (e.g. words), particularly when the set of symbols is of high cardinality. Because there is no prior notion of similarity that seems more preferable than others, a symbol is often encoded as a "one hot vector", i.e. a vector where the only non-zero element is the one corresponding to that symbol. This means that all symbols have a euclidean distance of 2 from all others. Hence the "one hot vector" representation captures well our ignorance about the similarities between symbols, but the dimensionality of the input space is then the number of different symbols, which can be very large (e.g. order of 10000 in NLP applications). Obviously, this becomes even worse when we consider sequences of such symbols, since the dimensionality of the "one hot vector" space for these sequences will then scale exponentially with the size of the symbol sequences.

A solution to this problem is to learn a vector representation $x(w) \in \mathcal{R}^{\mathcal{D}}$ for all symbols w in the set of possible symbols V. These "distributed representations" (Hinton, 1986) should contain information about w that is in a form helpful for the application considered. By having $D < |V|$, we effectively impose constraints on the way the information on w is stored, forcing the learning procedure for $x(w)$ to capture statistical similarities between the

different symbols. By mapping symbols in this continuous space, symbols that are nearby in that space are now more likely to have similar outputs when taken as input in a model. This implies that an algorithm that has learned a good output for a symbol w should now also have learned a reasonable output for nearby symbols in the space of distributed representations, hence diminishing the impact of the curse of dimensionality.

However, there is another impact of the curse of dimensionality that is not addressed by distributed representations as described above. Because $x(w)$ is only learned for the symbols in a finite, previously determined set V, generalization to the representations of new or unknown symbols that are not in V is not defined. Effectively, these symbols correspond to parts of the input space (and to dimensions of the input space with the "one hot vector" representation) with no training data. In the context of NLP applications, this is very unfortunate, given that there is virtually an infinite amount of new words that could be encountered because of variations in proper nouns or typographical errors.

Here, we propose a method to tackle this problem by learning $x(w)$ as a **function** of low-level or domain-knowledge features of w. This way, even in the case where a symbol was not present in the training set, it is possible to obtain a useful distributed representation of that symbol.

For the reminder of this paper, we are going to refer to the symbols w as words and to the symbol set V as the vocabulary, since most of the previous work and the experiments presented here focus on NLP tasks.

2 Distributed Representations of Words

In the context of Natural Language Processing (NLP), distributed representations have been used in many different applications, such as language modeling (Bengio et al., 2003), (Xu, Emami and Jelinek, 2003), speech recognition (Schwenk and Gauvain, 2002), information retrieval (Keller and Bengio, 2005) and PP-attachment resolution (Takahashi et al., 2001). Though the concept of distributed representations has mostly been used in feed-forward neural networks, it is flexible enough to be adapted to other types of models, e.g. hierarchical mixture of experts (Blitzer et al., 2005) or any model that can be trained using a gradient-based algorithm.

We define a distributed representation $x(w)$ of a word w simply as a vector representation in $\mathcal{R}^{\mathcal{D}}$. Though one might want to bound the components of $x(w)$, e.g. by considering these components as outputs of neurons, this is not necessary.

Also, if one is interested in a vector (e.g. sequence) of symbols $\vec{w} = (w_1, w_2, \ldots, w_n)'$, the associated vector representation is simply defined as the concatenation of the distributed representations:

$$x(\vec{w}) = (x(w_1)', x(w_2)', \ldots, x(w_n)')'$$

Note that, if one imposes $x(w)$ to be the "one hot vector" representation of w, then the space of $\vec{w}$ now only scales linearly with the size D of V, instead of exponentially, but this is still too large if D is big. By using distributed representations, we can control the size of $x(w)$ and $x(\vec{w})$ with the value of D.

Distributed representations can be learned in two ways: unsupervised or supervised manner. Here, we review some of the past proposals within those two frameworks.

2.1 Unsupervised Learning of Distributed Representations

Many different dimensionality reduction algorithms have been proposed in the past few years. These algorithms necessitate to define a high dimensional representation of the words (as for Principal Component Analysis (Jolliffe, 1986) and Locally Linear Embedding (Roweis and Saul, 2000)), a graph structure between words (as for ISOMAP (Tenenbaum, de Silva and Langford, 2000)), or simply a notion of similarity or dissimilarity between words (as for Multidimensional Scaling (Cox and Cox, 1994)).

A well known unsupervised method to learn distributed representations for words is Latent Semantic Analysis (Deerwester et al., 1990), which is a particular case of MDS when the similarity between two words w_1 and w_2 is defined as the number of pairs (w_1, w_2) that can be found within different documents.

In (Blitzer et al., 2005), the high dimensional representation of a word w_i is the vector of empirical conditional probabilities that word w_j will follow w_i, computed from a text corpus. Then, a slightly different variant of Semidefinite Embedding (Weinberger, Sha and Saul, 2004) is applied to obtain a distance matrix between words, which can be used by MDS to obtain a lower dimensional representation of these words. The variant of SDE is a semidefinite program that preserves only local distances between the high dimensional representations of words, not the angles.

The main advantage of unsupervised learning of distributed representations is that labeled text is not needed, hence the large and publicly available unlabeled corpora can be used. On the other hand, a definition of the high dimensional representation or similarity metric of words needs to be chosen, and the consequence of that choice on the performance for a particular task (e.g. POS tagging, word sense disambiguation, language modeling) is not necessarily clear. This is why it is usually used as an initialization stage before a supervised learning phase ((Bengio et al., 2003), (Blitzer et al., 2005)).

2.2 Supervised Learning of Distributed Representations

When we know that the distributed representations are going to be used for a particular task, it is usually more effective to learn or tune them in a supervised manner. We focus here on the case where the distributed representations are used as the input of a feed-forward neural network, since we are going to propose an extension and report results for that case.

Imagine that we are given a training set $\mathcal{T} = \{(\vec{w}_1, y_1), \ldots, (\vec{w}_n, y_n)\}$ corresponding to a particular task, where $\vec{w}_t = (w_1, \ldots, w_c)'$ is referred to as the "context" and y_t, the target to be predicted. In the case of POS tagging, $\vec{w}_t$ would contain a word at position t in a text with its surrounding words, and y_t would be the syntactic class of w_t. In the case of language modeling, $\vec{w}_t$ would contain c words before the t^{th} word of a text, and y_t would be w_t.

The idea is to learn the parameters of a neural network and the distributed representations simultaneously. The architecture of the neural network is depicted in figure 1. We use hyperbolic tangent as the activation function of the hidden layer nodes, and the softmax function to get a probability vector of the target items in output.

Assuming that the training cost to optimize is differentiable with respect to the output of the neural network, it is straightforward to obtain the gradient of the cost with respect to the parameters of the neural network and the distributed representations. We can then use a gradient-based optimization procedure (such as stochastic gradient descent) to train the network and learn the distributed representations and the neural net parameters simultaneously.

The advantage of learning the representations that way is that we are sure that they will contain information about words that is useful to the task considered. Of course, the drawback is that labeled data is needed, except in the case of language modeling.

2.3 Generalization to New Words

In the case of unsupervised methods, (Bengio et al., 2004) make some proposals for out-of-sample generalization of the embedding learned by dimensionality reduction techniques. These proposals though don't change the fact that the representations may not be useful for the task considered. Also, the computations involved in these proposals scale with the cardinality of the predefined set of words.

In the supervised case, the only proposition we are aware of was made in (Bengio et al., 2003). When a new word w_t is encountered at position t, its distributed representation becomes:

$$E\left[C(w_t)|h_t\right] = \sum_{w \text{ in short list}} C(w)P(w_t = w|h_t)$$

where $C(w)$ is the representation given by the table look-up for w, h_t is the vector of words preceding w_t and $P(w_t = w|h_t)$ is given by the neural network. The short list is composed of the most probable next words given h_t according to a standard smoothed trigram model. Besides the fact that this method can only be applied to language modeling, the representations of new words won't make much sense if the terms in the expectation with a significant value of $P(w_t = w|h_t)$ have very different values of $C(w)$.

In general, a lot of attention in the field of NLP is dedicated to the generalization of systems to new words (see (Toutanova and Manning, 2000), (Nakagawa, Kudoh and Matsumoto, 2001)). It is the main motivation behind Named Entity Recognition systems (see CoNLL-2002 and CoNLL-2003 shared tasks) that permit to detect locations, persons or organizations references, which are often words or expressions that were never seen before. It then makes sense to put some efforts in adapting supervised distributed representations to the case of new words.

3 Proposed Approach

The fact that it is not clear how to generalize supervised distributed representations to new words comes from the choice to ignore other types of information about these words. However, there are several sources of information that are available:

- low-level information, e.g. characters, syllables or morphological form of a word;
- domain-knowledge information, e.g. position of a word in a thesaurus, ontology or dictionary (e.g. WordNet, FrameNet);
- statistical information, e.g. frequency counts of pairs of words in sentences, paragraphs or documents.

In order to generalize supervised distributed representations to unknown words, we propose to model the distributed representation $x(w)$ of a word w as a **function** of the information available about that word. This function will consider different types of information about w among those mentioned above, and will take the form of a feature vector. In the case of a feed-forward neural network, the architecture of figure 1 becomes as depicted in figure 2.

The original feed-forward neural network of figure 1 is really a particular case of our proposed approach, where the function $x(w)$ is simply a linear application on the "one hot vector" feature vector of w.

Note that any model that can be trained using a gradient-based algorithm and that takes a real-valued vector as input can be easily extended to make use of our proposal.

4 Experiments

We present here experiments on a POS tagging task. The task consists of assigning the correct syntactic category to a word, given that word and its surrounding words in the text. We used the Wall Street Journal portion of the Penn TreeBank II corpus. Sections 19 to 21 and sections 22 to 24 were used as our validation and test corpora, respectively. The different corpora were processed the following way:

- sentences were separated, in order to not consider surrounding words from different sentences. When the context overlapped on a different sentence, the words out of the sentence were replaced with a pseudo word we call "out of vocabulary". The only exception made was for ".", in which case all words in the sentence on the left and on the right were included;

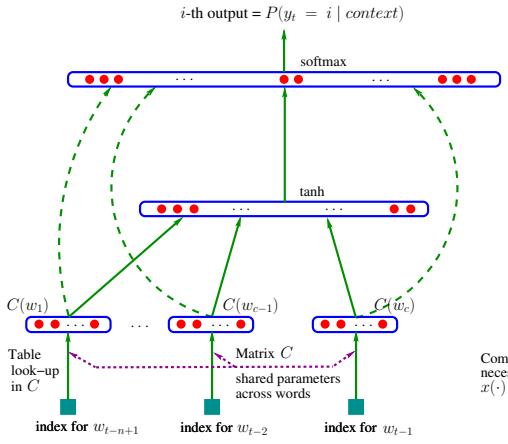

Figure 1: Feed-Forward Neural Network with distributed representations stored in a look-up table.

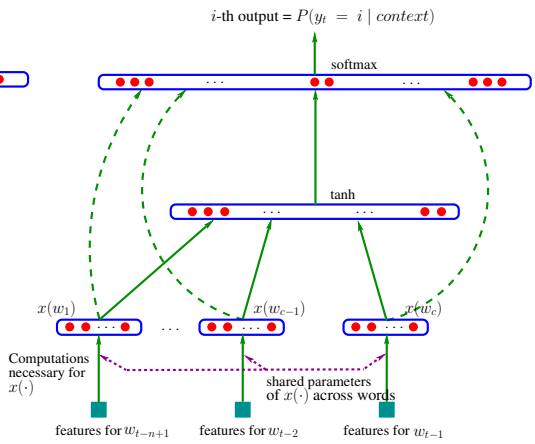

Figure 2: Feed-Forward Neural Network with distributed representations as a function.

- "." is not considered as part of a sentence;
- when the target tag was ambiguous (i.e. a list of more than one tag was given as target), we only considered the first tag in the list;
- words were not put to lower case.

We tested the three following models:

1. neural network, as depicted in figure 1, with the number of hidden units as an hyper-parameter. We defined the case of no hidden units as a log-linear model on the distributed representations. We refer to this model as **NNet-Table**;
2. neural network with distributed representations as a function of the words, as depicted in figure 2. We used the same hyper-parameters as the first model. The function $x(w)$ is linear in the features of w. We refer to this model as **NNet-Func**;
3. log-linear model (or maximum entropy model), which is widely used in NLP systems, directly on the input feature vector. We refer to this model as **Log-Lin**.

For the first model, we constructed the predefined set of words (or dictionary) V by taking the words in the training set that appeared at least three times. In the validation and test corpora, we replaced words that were not in V with the "out of vocabulary" pseudo word, for which NNet-Table also learns a distributed representation.

The second and third models used the common word features for NLP systems, that is a set of features for the "whole word" (not lower cased), and for 4-character prefixes and suffixes. Also, four binary features indicating if a word starts with a capital letter, only contains capital letters, contains only digits or contains only digits and the characters "." and ",". The first three set of features were filtered so that only the features activated at least three times in the training set were considered. We did not use information about the previous tags, mostly because we want to compare the generalization for new words based only on information about these words, not based on the information about surrounding tags.

Table 1: Test error percentages on the POS tagging task. The training set contains the sections 00 to 18 of the Wall Street Journal portion of Penn TreeBank II. The differences between the test errors are all statistically significant under a T-test.

Algorithm	**Global Test Error**	**Test Error on New Words**
NNet-Table	4.49%	35.53%
Log-Lin	3.47%	14.80%
NNet-Func	**3.32**%	**14.21**%

The distributed representation dimensionality D was set to 20 in all experiments. We used stochastic gradient descent on the negative log-likelihood of the targets to train all models, using early stopping on a validation set. The learning rate and decrease constant were also chosen so to optimize the error rate on the validation set, as were the number of surrounding words considered at the left and right of a word to tag.

To improve generalization performance, we also used an L2-norm penalization on the activated weights for the 3 models.

The results are reported in table 1. We can see that NNet-Func obtains the best performance, both in global and "new words" test error. The "new words" include all words for which no features of type "whole word" in the feature set is activated. It also correspond to the words for which NNet-Table The closest model to NNet-Func in terms of test error is the Log-Lin model. As expected, the NNet-Table model does a lot worse, especially for new words, since NNet-Table does not exploit the prior similarity induced by our word features set.

For both NNet-Table and NNet-Func, adding neurons did not provide any improvement in error. This makes the improvement on Log-Lin even more surprising, since a log-linear model already has low capacity. In order to try to explain this, we looked at the training errors obtained by NNet-Func and Log-Lin. NNet-Func has much higher training error (2.52%) than Log-Lin (1.85%), which indicates that Log-Lin might have too much capacity for the input space considered. This also supports the claim that learning distributed representations corresponds to a better model of language, even in the log-linear case. Also, with distributed representations of size 20, the optimal number of words on the left and right of the word to tag is respectively 5 and 2 for NNet-Func, and 2 and 1 for Log-Lin. This indicates that NNet-Func is able to make use of more context words than Log-Lin, which is a property that may be very useful for other applications.

Also, we varied the size of the training set in order to observe its impact on the relative performances of the different models. The results are showed in figure 3. We did not include the test errors for new words under the NNet-table model simply because they are very much higher than those of the other models. Besides the difference on the test errors for new words with the 00-14 training set sections, the differences between the test errors are all statistically significant under a T-test. It should noted that the context of words was limited to a maximum of three words to the left or to the right of the word to tag, simply to reduce the number of possible configurations of the context to test. It is hence very much likely that better results could be obtained with NNet-Func by allowing the context to go up to five words around the word to tag.

Finally, in order to gain more insight into what information was captured by our proposed model, we show in table 2 different words with their five nearest neighbors in the space of distributed representations.

We can see that the distributed representations captured many different types of information, mostly syntactic information (see neighbors of "will", "and" and "make" for example) but also semantic information (see neighbors of "researchers", "Friday", "December" and "Italian" for example). There are some nearest neighbors that can seem odd, e.g. "Chamber" is a nearest neighbor of "December", but note that the representations should contain information that is useful for POS tagging and that is biased by the choice of word features ("December" and "Chamber" both end in "mber").

Figure 3: Relationship between the size of the training set and the test performance of the compared models.

We believe that NNet-Table would have learned similar distributed representations. But, because it is not using the low-level word features, it would probably not have grouped words that start with a capital letter together (see "Eight" and its neighbors). Also, as noted earlier, NNet-Table has considerably higher test errors.

Encouraged by these results, we decided to add more low-level features in input in order to enrich the distributed representations. We added features for prefix and suffix of length 1 to 3 and 5. We used the hyper-parameters chosen in the preceeding experiments of table 1, but we allowed the distributed representations to be of size 30 instead of 20, which gave better validation error. On the test set, NNet-Func now obtains **3.296%** global error and an impressive **11.82%** error on new words, compared to 3.42% global error and only 12.49% error on new words for Log-Lin with these new features.

Usually, a different definition of "new word" is used, that is words which have been cut off the vocabulary V because they did not appear at least 3 times in the training corpora is not considered as a "new word", contrary to the results reported so far. We computed the test error for new words under this new definition for NNet-Func and Log-Lin, and we obtained respectively 13.23% and 13.43%.

5 Conclusion and Future Work

We have presented an extension of supervised distributed representations learning which permits generalization to new words and provides significant improvement in accuracy over a state of the art log-linear model in the context of a POS tagging system. Hence, we have proposed a model which combines the capacity of neural networks to learn representations of words (which could be used in other contexts, such as language modeling) and the discriminative power of log-linear models in NLP.

Though POS tagging systems usually optimize the whole sequence of POS tags, e.g. with dynamic programming, the model we tested only considered the word to tag and its surrounding words in its prediction and made the predictions of the POS tags independently. Hence, we believe that an extension of Conditional Random Fields (Lafferty, McCallum and Pereira, 2001) that would use distributed representations would be a natural next step. The implementation should not be problematic, since the training of CRFs can be done with gradient-based algorithms. Then, distributed representations would also be needed for POS tags, and we might expect that it will be possible to introduce higher order dependance in the POS tags and obtain better accuracy, as it has been observed here with word distributed representations.

Table 2: Nearest neighbors in the space of distributed representations for words in V. The neighbors are in decreasing order of distance.

Word	Nearest Neighbors	**Word**	Nearest Neighbors
will	could should would might must	Consolidated	Integrated Illustrated Automated Associated Federated
the	any an The a An	once	meanwhile afterward desperately beforehand afterwards
a	an any no all the	caused	allowed received requested pushed determined
and	but or nor & yet	researchers	brothers scientists teachers philosophers adjusters
1/2	1/4 3/4 7/8 5/8 3/8	more	closer earlier higher faster tougher
was	got had did ran met	Eight	Seven Twenty Four Nine Three
it	he she him us me	make	want take give allow reduce
30	44 65 61 900 55	than	because after about from without
.	? ! 've our my	with	from between without against over
–	; : ... - 'n'	latest	fastest greatest smartest hottest brightest
Mr.	Ms. Co. J. Fed Dow	England	Scotland Lilly Minneapolis Georgia Milton
The	An All Any A No	company	inflation syndication competence market recession
up	off out down on over	Italian	Brazilian Jovian Secret Argentine Nicaraguan
its	his our my her your	Wednesday	Thursday Sunday Holiday Monday Paul
new	old big few hot bad	Friday	Thursday Holiday Wednesday Monday Solidarity
This	Each Every Another Some These	first-half	half-hour one-half half-dozen second middle
We	You He She It I	eight-year	five-year 30-year seven-year three-year 30-share
1950s	1960s 1980s 1990s 1930s '80s	second	third fifth guilty one-fifth fourth
seven	four five nine eight three	December	November September October Chamber April
		Ohio-based	Minneapolis-based Calgary-based Denver-based Miami-based Chicago-based

Another interesting extension would consist of initializing the distributed representations by first training a neural network language model on large unlabeled corpora. By doing this, the distributed representation predictor can be trained on some words which would have been unknown otherwise. Preliminary results seem to indicate that a POS tagger performance can be increased that way, which encourages us to pursue this avenue.

References

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N., and Ouimet, M. (2004). Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press.

Blitzer, J., Weinberger, K., Saul, L., and Pereira, F. (2005). Hierarchical distributed representations for statistical language modeling. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA.

Cox, T. and Cox, M. (1994). *Multidimensional Scaling*. Chapman & Hall, London.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Hinton, G. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986. Lawrence Erlbaum, Hillsdale.

Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

Keller, M. and Bengio, S. (2005). A neural network for text representation. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Biological Inspirations, ICANN, Lecture Notes in Computer Science*, volume LNCS 3697, pages 667–672.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML'2001*.

Nakagawa, T., Kudoh, T., and Matsumoto, Y. (2001). Unknown word guessing and part-of-speech tagging using support vector machines. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 325–331, Tokyo, Japan.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 765–768, Orlando, Florida.

Takahashi, N., Motoki, M., Shimazu, Y., Tomiura, Y., and Hitaka, T. (2001). Pp-attachment ambiguity resolution using a neural network with modified fgrep method. In *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks*, Tokyo.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP/VLC 2000*, pages 63–70.

Weinberger, K. Q., Sha, F., and Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada.

Xu, P., Emami, A., and Jelinek, F. (2003). Training connectionist models for the structured language model. In *Empirical Methods in Natural Language Processing, EMNLP'2003*.