INCORPORATING BOTH DISTRIBUTIONAL AND RELATIONAL SEMANTICS IN WORD REPRESENTATIONS

Daniel Fried∗
Department of Computer Science
University of Arizona
Tucson, Arizona, USA
dfried@email.arizona.edu

Kevin Duh
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara, JAPAN
kevinduh@is.naist.jp

ABSTRACT
We investigate the hypothesis that word representations ought to incorporate both distributional and relational semantics. To this end, we employ the Alternating Direction Method of Multipliers (ADMM), which flexibly optimizes a distributional objective on raw text and a relational objective on WordNet. Preliminary results on knowledge base completion, analogy tests, and parsing show that word representations trained on both objectives can give improvements in some cases.

1 INTRODUCTION
We are interested in algorithms for learning vector representations of words. Recent work has shown that such representations can capture the semantic and syntactic regularities of words (Mikolov et al., 2013a) and improve the performance of various Natural Language Processing systems (Turian et al., 2010; Wang & Manning, 2013; Socher et al., 2013a; Collobert et al., 2011).

Although many kinds of representation learning algorithms have been proposed so far, they are all essentially based on the same premise of distributional semantics (Harris, 1954). For example, the models of (Bengio et al., 2003; Schwenk, 2007; Collobert et al., 2011; Mikolov et al., 2013b; Mnih & Kavukcuoglu, 2013) train word representations using the context window around the word. Intuitively, these algorithms learn to map word representations of similar context to nearby points in vector space.

However, distributional semantics is by no means the only theory of word meaning. Relational semantics, exemplified by WordNet (Miller, 1995), defines a graph of relations such as synonymy and hyponymy (Cruse, 1986) between words, reflecting our world knowledge and psychological predispositions. For example, a relation like “dog is-a mammal” describes a precise hierarchy that complements the distributional similarities observable from corpora.

We believe both distributional and relational semantics are valuable for word representations, and investigate combining these approaches into a unified representation learning algorithm based on the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011). Its advantages include (a) flexibility in incorporating arbitrary objectives, and (b) relative ease of implementation. We show that ADMM effectively optimizes the joint objective and present preliminary results on several tasks.

2 DISTRIBUTIONAL AND RELATIONAL OBJECTIVES
Distributional Semantics Objective: We implement distributional semantics using the Neural Language Model (NLM) of (Collobert et al., 2011). Each word $i$ in the vocabulary is associated with a $d$-dimensional vector $w_i \in \mathbb{R}^d$, the word’s embedding. An $n$-length sequence of words $(i_1, i_2, \ldots, i_n)$ is represented as a vector $x$ by concatenating the vector embeddings for each word, $x = [w_{i_1}; w_{i_2}; \ldots; w_{i_n}]$. This vector $x$ is then scored by feeding it through a two-layer neural network with $h$ hidden nodes: $S_{NLM}(x) = u^\top(f(Ax + b))$, where $A \in \mathbb{R}^{h \times (nd)}, b \in \mathbb{R}^h, u \in \mathbb{R}^h$.

∗Currently at the University of Cambridge.
are network parameters and \( f \) is the sigmoid \( f(t) = 1/(1 + e^{-t}) \) applied element-wise. The model is trained using noise contrastive estimation (NCE) (Mnih & Kavukcuoglu 2013), where training text is corrupted by random replacement of random words to provide an implicit negative training example, \( x_c \). The hinge-loss function, comparing positive and negative training example scores, is:

\[
L_{NLM}(x, x_c) = \max(0, 1 - S_{NLM}(x) + S_{NLM}(x_c)) \tag{1}
\]

The word embeddings, \( w \), and other network parameters are optimized with backpropagation using stochastic gradient descent (SGD) over n-grams in the training corpus.

**Relational Semantics Objective**: We investigate three different objectives, each modeling relations from WordNet. The Graph Distance loss, \( L_{GD} \), enforces the idea that words close together in the WordNet graph should have similar embeddings in vector space. First, for a word pair \((i, j)\), we define a pairwise word similarity \( WordSim(i, j) \) as the normalized shortest path between the words’ synonym sets in the WordNet relational graph [Leacock & Chodorow, 1998]. Then, we encourage the cosine similarity between their embeddings \( v_i \) and \( v_j \) to match that of \( WordSim(i, j) \):

\[
L_{GD}(i, j) = \left( \frac{v_i \cdot v_j}{||v_i||_2 ||v_j||_2} - [a \times WordSim(i, j) + b] \right)^2 \tag{2}
\]

where \( a \) and \( b \) are parameters that scale \( WordSim(i, j) \) to be of the same range as the cosine similarity. Training proceeds by SGD: word pairs \((i, j)\) are sampled from the WordNet graph, and both the word embeddings \( v \) and parameters \( a, b \) are updated by gradient descent on the loss function.

A different approach directly models each WordNet relation as an operation in vector space. These models assign scalar plausibility scores to input tuples \((v_l, R, v_r)\), modeling the plausibility of a relation of type \( R \) between words \( v_l \) and \( v_r \). In both of the relational models we consider, each type of relationship (for example, synonymy or hypernymy) has a distinct set of parameters used to represent the relationship as a function in vector space. The TransE model of [Bordes et al. 2013] represents relations as linear translations: if the relationship \( R \) holds for two words \( v_l \) and \( v_r \), then their embeddings \( v_l, v_r \in \mathbb{R}^d \) should be close after translating \( v_l \) by a relation vector \( R \in \mathbb{R}^d \):

\[
S_{TransE}(v_l, R, v_r) = -||v_l + R - v_r||_2 \tag{3}
\]

Socher et al. [2013b] introduce a Neural Tensor Network (NTN) that models interaction between embeddings using tensors and a non-linearity function. The scoring function for a input tuple is:

\[
S_{NTN}(v_l, R, v_r) = U^T f \left( v_l^T W_R v_r + V_R \frac{v_l}{v_r} + b_R \right) \tag{4}
\]

where \( U \in \mathbb{R}^h, W_R \in \mathbb{R}^{d \times h} \), \( V_R \in \mathbb{R}^{h \times 2d} \) and \( b_R \in \mathbb{R}^h \) are parameters for relationship \( R \). As in the NLM, parameters for these relational models are trained using NCE (producing a noisy example for each training example by randomly replacing one of the tuples’ entries) and SGD, using the hinge loss as defined in Eq. [1] with \( S_{NLM} \) replaced by the \( S_{TransE} \) or \( S_{NTN} \) scoring function.

**Joint Objective Optimization by ADMM**: We now describe an ADMM formulation for joint optimization of the above objectives. Let \( w \) be the set of word embeddings \( \{w_1, w_2, \ldots, w_N\} \) for the distributional objective, and \( v \) be the set of word embeddings \( \{v_1, v_2, \ldots, v_{N'}\} \) for the relational objective, where \( N' \) and \( N'' \) are the vocabulary size of the corpus and WordNet, respectively. Let \( I \) be the set of \( N \) words that occur in both. Then we define a set of vectors \( y = \{y_1, y_2, \ldots, y_N\} \), which correspond to Lagrange multipliers, to penalize the difference \( (w_i - v_i) \) between sets of embeddings for each word \( i \) in the joint vocabulary \( I \), producing a Lagrangian penalty term:

\[
L_P(w, v) = \sum_{i \in I} \left( y_i^T (w_i - v_i) \right) + \frac{\rho}{2} \sum_{i \in I} (w_i - v_i)^T (w_i - v_i) \tag{5}
\]

In the first term, \( y \) has same dimensionality as \( w \) and \( v \), so a scalar penalty is maintained for each entry in every embedding vector. This constrains corresponding \( w \) and \( v \) vectors to be close to each other. The second residual penalty term with hyperparameter \( \rho \) is added to avoid saddle points; \( \rho \) can be viewed as a step-size during the update of \( y \).

This augmented Lagrangian term (Eq. [5]) is added to the sum of the loss terms for each objective (Eq. [1] and Eq. [2]). Let \( \theta = (u, A, b) \) be the parameters of the distributional objective, and \( \phi \) be the parameters of the relational objective. The final loss function we optimize becomes:

\[
L = L_{NLM}(w, \theta) + L_{GD}(v, \phi) + L_P(w, v) \tag{6}
\]
Figure 1: Analysis of ADMM behavior by training iteration, for varying $\rho$. Left: Joint loss, $L_{NLM} + L_{GD}$, on the training data. Right: Normalized residual magnitude, averaged across embeddings.

<table>
<thead>
<tr>
<th>Knowledge Base</th>
<th>NLM</th>
<th>GD</th>
<th>GD+NLM</th>
<th>TransE</th>
<th>TransE+NLM</th>
<th>NTN</th>
<th>NTN+NLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Test</td>
<td>42</td>
<td>41</td>
<td>41</td>
<td>37</td>
<td>38</td>
<td>36</td>
<td>41</td>
</tr>
<tr>
<td>Parsing</td>
<td>76.03</td>
<td>75.90</td>
<td>76.18</td>
<td>75.86</td>
<td>76.01</td>
<td>75.85</td>
<td>76.14</td>
</tr>
</tbody>
</table>

Table 1: Results summary: Accuracy on knowledge base completion, MaxDiff accuracy on Analogy Test, and Label Arc Score Accuracy on Dependency Parsing for single- and joint-objective models.

The ADMM algorithm proceeds by repeating the following three steps until convergence:

1. Perform SGD on $w$ and $\theta$ to minimize $L_{NLM} + L_P$, with all other parameters fixed.
2. Perform SGD on $v$ and $\phi$ to minimize $L_{GD} + L_P$, with all other parameters fixed.
3. For all embeddings $i$ corresponding to words in both the n-gram and relational training sets, update the constraint vector $y_i := y_i + \rho(w_i - v_i)$.

Since $L_{NLM}$ and $L_{GD}$ share no parameters, Steps (1) and (2) can be optimized easily using the single-objective NCE and SGD procedures, with additional regularization term $\rho(w_i - v_i)$.

3 PRELIMINARY EXPERIMENTS & DISCUSSIONS

The distributional objective $L_{NLM}$ is trained using 5-grams from the Google Books English corpus containing over 180 million 5-gram types. The top 50k unigrams by frequency are used as the vocabulary, and each training iteration samples 100k n-grams from the corpus. For training $L_{GD}$, we sample 100k words from WordNet and compute the similarity of each to 5 other words in each ADMM iteration. For training $L_{TransE}$ and $L_{NTN}$, we use the dataset of Socher et al. (2013b), presenting the entire training set of correct and noise-contrastive corrupted examples one instance at a time in randomized order for each iteration.

We first provide an analysis of the behavior of ADMM on the training set, to confirm that it effectively optimizes the joint objective. Fig. 1(left) plots the learning curve by training iteration for various values of the $\rho$ hyperparameter. We see that ADMM attain a reasonable objective value relatively quickly in 100 iterations. Fig. 1(right) shows the averaged difference between the resulting sets of embeddings $w$ and $v$, which decreases as desired.

Next, we compare the embeddings learned with different objectives on three standard benchmark tasks (Table 1). First, the Knowledge Base Completion task (Socher et al. 2013b) evaluates the models’ ability to classify relationship triples from WordNet as correct. Triples are scored using the relational scoring functions (Eq.3 and 4) with the learned model parameters. The model uses a development set of data to determine a plausibility threshold, and classifies triples with a higher score than the threshold as correct, and those with lower score as incorrect. Secondly, the SemEval2012 Analogy Test is a relational word similarity task similar to SAT-style analogy questions (Jurgens et al., 2012). Given a set of four or five word pairs, the model selects the pairs that most and least represent a particular relation (defined by a set of example word pairs) by comparing the cosine similarity of the vector difference between words in each pair. Finally, the Dependency Parsing task on the SANC2012 data (Petrov & McDonald, 2012) evaluates the accuracy of parsers trained on news domain adapted for web domain. We incorporate the embeddings as additional features in

$^1$Berkeley distribution: tomato.banatao.berkeley.edu:8080/berkeylem_binaries/

$^2$The reason for the peak around iteration 50 in Fig. 1 is that the embeddings begin with similar random initializations, so initially differences are small; as ADMM starts to see more data, $w$ and $v$ diverge, but converge eventually as $y$ become large.
a standard maximum spanning tree dependency parser to see whether embeddings improve generalization of out-of-domain words. The evaluation metric is the labeled attachment score, the accuracy of predicting both correct syntactic attachment and relation label for each word.

For both Knowledge Base and Parsing tasks, we observe that joint objective generally improves over single objectives: e.g. TransE+NLM (83.10%) > TransE (82.87%) for Knowledge Base, GD+NLM (76.18%) > GD (75.90%) for Parsing. The improvements are not large, but relatively consistent. For the Analogy Test, joint objectives did not improve over the single objective NLM baseline. We provide further analysis as well as extended descriptions of methods and experiments in a longer version of the paper here: [http://arxiv.org/abs/1412.4369](http://arxiv.org/abs/1412.4369).

ACKNOWLEDGMENTS

This work is supported by a Microsoft Research CORE Grant and JSPS KAKENHI Grant Number 26730121. D.F. was supported by the Flinn Scholarship during the course of this work. We thank Haixun Wang, Jun’ichi Tsujii, Tim Baldwin, Yuji Matsumoto, and several anonymous reviewers for helpful discussions at various stages of the project.

REFERENCES


