1 Learning Algorithm of L-NDMV

The objective function of learning L-NDMV is the log-likelihood $L(\Theta)$ of the training sentences $X = \{x_1, x_2, ..., x_n\}$, where the parameters of the neural network are denoted by $\Theta$.

$$L(\Theta) = \sum_{a=1}^{n} \log P(x_a; \Theta)$$

This objective function can be optimized using the hard expectation-maximization algorithm. The hard E-Step uses the current model parameterized by $\Theta$ to parse each training sentence $x_i$ and then calculates the counts of all the grammar rules in the best parse of $x_i$, represented by $e_c(x_i)$ for the CHILD rule $c$, $e_d(x_i)$ for the DECISION rule $d$ and $e_r(x_i)$ for the ROOT rule $r$ respectively. Let $\phi(X)$ be the set of counts of all the grammar rules used in the best parses of all the training sentences in $X$. The M-Step optimizes the following objective function:

$$l(\Theta; \phi(X)) = \sum_{i=1}^{n} \left( \sum_{c} e_c(x_i) \log p_c \right) + \sum_{d} e_d(x_i) \log p_d + \sum_{r} e_r(x_i) \log p_r$$

This learning method when applied to L-NDMV with a large training corpus is very time-consuming. We propose two improvements to the learning method to achieve significant speedup. First, at each E-step we calculate the grammar rule counts from a different batch of sentences instead of from the whole training corpus and then train the neural network using only these counts. Second, unlike in the NDMV approach, we no longer train a new neural network from scratch at each M-step but instead train the same network network across EM iterations. Together, these two improvements speed up training by more than 50 times with almost no drop in learning accuracy. Algorithm 1 outlines our learning algorithm of L-NDMV.

Our algorithm resembles online EM (Liang and Klein, 2009) except that at each EM iteration we utilize counts collected from only the current batch of training sentences. Because each count of grammar rule usage becomes a training sample for the neural network, training the neural network with counts collected from all the sentences, as in standard online EM, would beat the purpose of online learning and significantly slow down training.

2 Visualization of the Learned Word Vectors

To analyze what information is learned to be encoded in the input continuous representation of head words, we used the t-SNE algorithm (Van der Maaten and Hinton, 2008) to map the word vectors learned by our approach onto a 2D plane in Figure
In order to avoid interference of the word vector initialization method, we used random initialization instead. Interestingly, words that are close to each other in the figure (and hence have similar word vectors) are not only syntactically similar, but also semantically similar. For instance, three noun abbreviations “inc.”, “co.” and “corp.” have very similar coordinates in the figure. This is because semantically similar words typically have similar contexts in the training corpus and such similarity can be captured by our grammar induction approach. Lexical semantics, as captured here in word vectors, is beneficial to syntactic parsing when there are rare words or syntactic ambiguities (Le and Zuidema, 2015). For example, if from the training data it is learned that a dependency between “in” and “corp.” is very likely, then even if “in” and “co.” never co-occur in the training data, our approach would still learn the dependency relation between them.

## 3 The Dependency Rules of Chinese

When using the grammar induction method of Naseem et al. (2010) to initialize our model as described in section 2.3, we need to specify a set of prior dependency rules. The dependency rules employed by Naseem et al. include a set of universal dependency rules as well as a few English-specific rules. We replace the English-specific rules with a few Chinese-specific rules designed by ourselves.

### References


