This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

Distributed Feature Representations for Dependency Parsing

Wenliang Chen and Min Zhang (Member IEEE) and Yue Zhang

***Abstract*—This article presents an approach to automatically learning distributed representations for features to address the feature sparseness problem for dependency parsing. Borrowing terminologies from word embeddings, we call the feature representation feature embeddings. In our approach, the feature embeddings are inferred from large amounts of auto-parsed data. Firstly, the sentences in raw data are parsed by a baseline system and we obtain dependency trees. Then, we represent each model feature using the surrounding features on the dependency trees. Based on the representation of surrounding context, we proposed two learning methods to infer feature embeddings. Finally, based on feature embeddings, we present a set of new features for graph-based dependency parsing models. The new parsers can not only make full use of well-established hand-designed features but also benefit from the hidden-class representations of features. Experiments on the standard Chinese and English data sets show that the new parser achieves significant performance improvements over a strong baseline.**

I. Introduction

In recent years, discriminative supervised models have achieved much progresses in dependency parsing [1]. The discriminative models typically use millions of features generated from a small set of training data. This setting has shown strong discriminative power in previous studies [2], [3], [4], [5]. However, binary features extracted from a limited size training data (typically less than fifty thousands of sentences for dependency parsing) suffer from data sparseness: for features that are rare in the labeled training data, the corresponding model parameters could be poorly estimated.

Another limitation on features is that many are typically derived by (manual) combination of atomic features. For example, given the head word (w_h) and part-of-speech tag (p_h), dependent word (w_d) and part-of-speech tag (p_d), and the label (l) of a dependency arc, state-of-the-art dependency parsers can have the combined features: $[w_h; p_h]$, $[w_h; p_h; w_d; p_d]$, $[w_h; p_h; w_d]$, and so on, in addition to the atomic features: $[w_h]$, $[p_h]$, etc. Such combination is necessary for high accuracies because the dominant approach uses linear models. However, the correlations between features are still unknown.

We tackle the above issues by borrowing solutions from word representations, which have been intensely studied in the NLP community [6]. In particular, distributed representations of words have been used for many NLP problems, which represent a word by information from the words it frequently co-occurs with [7], [8], [9], [10], [11]. The representation can be learned from large amounts of raw sentences, and hence used to reduce OOV rates in test data. In addition, since the representation of each word carries information about its context words, it can also be used to calculate word similarity [12], or used as additional semantic features [13].

In this article, we move beyond word embeddings and consider the vector representation of features for discriminative linear dependency parsing models. Our target is to learn distributed feature representations (referred to as feature embeddings) that can not only make full use of well-established hand-designed features but also benefit from hidden representations of features. The idea behind word embeddings is the distributional hypothesis in linguistics, which states that words appearing in similar contexts tend to have similar meanings [14]. Similarly, we believe that features that occur in similar contexts on dependency trees tend to share common properties.

Compared with the task of learning word embeddings, the task of learning feature embeddings is more difficult because the size of features is much larger than the vocabulary size and tree structures are more complex than word sequences. This requires us to find an effective inference algorithm to learn feature embeddings. [11] and [12] introduce efficient models to learn high-quality word embeddings from large amount of raw text data, implemented in word2vec.[1] We adapt their methods to learn feature embeddings. In our approach, we first propose a novel approach to represent features. We use two learning models, similar to the CBOW and Skip-gram models proposed by [12], to infer distributed representations for features. Based on the feature embeddings, a set of new features are designed and incorporated into the parsing models.

To demonstrate the effectiveness of the feature embeddings, we apply them to a graph-based parsing model [15]. We conduct experiments on the standard data sets from the Penn English Treebank [16] and the Chinese Treebank Version 5.1 [17]. The results indicate that our proposed approach significantly improves the accuracy.

This article is a significant extension of a conference version [18]. We add a new learning model to infer feature embeddings, new experimental results, a comprehensive description of the parsing models, and more details about our method.

The rest of this article is organized as follows. Section II introduces the background of graph-based dependency parsing.

Correspondence should be sent to M. Zhang.

W. Chen and M. Zhang are with School of Computer Science and Technology, Soochow University, Suzhou, China (e-mail: chenwenliang@gmail.com; zhangminmt@hotmail.com)

Y. Zhang is with Singapore University of Technology and Design, Singapore (e-mail:yue_zhang@sutd.edu.sg)

[1] https://code.google.com/p/word2vec/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

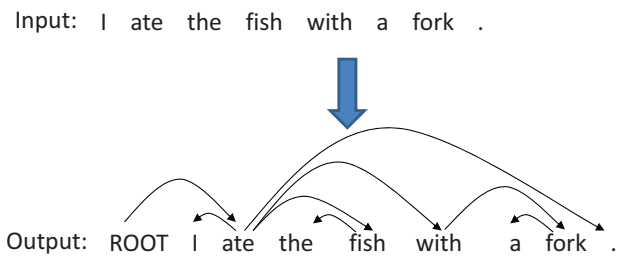

Fig. 1. Example for dependency parsing task

Section III describes the two models to infer feature embeddings. Section IV describes the parser with the embedding-based features. Section V shows the implementation details of our systems. Section VI describes the experimental settings and reports the experimental results on the English and Chinese data sets. Section VII discusses related work. Finally, in Section VIII we draw conclusions on the proposed approach.

II. Background of dependency parsing

In this section, we introduce the background of dependency parsing and build a baseline parser based on the graph-based parsing model proposed by [19].

A. Dependency parsing

Given an input sentence x, the task of dependency parsing is to build a dependency tree y. Figure 1 shows an example of the input and output of dependency parsing, where an arc between two words indicates a dependency relation between them, "ROOT" is an artificial root token inserted at the beginning of the sentence and is not to be a dependent of any other token in the sentence. For example, the arc between "ate" and "fish" indicates a dependency where "ate" is the head and "fish" is the dependent. The arc between "ROOT" and "ate" indicates that "ate" is the ROOT of the sentence.

x is denoted by $x = (w_0, w_1, ..., w_i, ..., w_m)$, where w_0 is ROOT and w_i refers to a word. We have a set of training data $D = (x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)$ to train a parser. The task of dependency parsing is to find y^* which has the highest score for x,

$$y^* = \arg\max_{y \in Y(x)} score(x, y)$$

where $Y(x)$ is the set of all the valid dependency trees for x.

For dependency parsing, there are two major models [20]: the transition-based model and graph-based model, which showed comparable accuracies for a wide range of languages [1], [4], [5], [21]. The main difference between the two models is on whether the parse tree is scored directly (i.e. graph-based) or indirectly via a sequence of transition actions (i.e. transition-based). In this article, we apply feature embeddings to a graph-based model.

B. Graph-based parsing model

In the graph-based model, we use an ordered pair $(w_i, w_j) \in y$ to define a dependency relation in tree y from word w_i to word w_j (w_i is the head and w_j is the dependent), and G_x to define a graph that consists of a set of nodes $V_x = \{w_0, w_1, ..., w_i, ..., w_m\}$ and a set of arcs (edges) $E_x = \{(w_i, w_j) | i \neq j, w_i \in V_x, w_j \in (V_x - \{w_0\})\}$. The parsing model of [19] searches for the maximum spanning tree (MST) in graph G_x. We denote $Y(G_x)$ as the set of all the subgraphs of G_x that are valid dependency trees [15] for sentence x.

We define the score of a dependency tree $y \in Y(G_x)$ to be the sum of the subgraph scores,

$$score(x, y) = \sum_{g \in y} score(x, g) \quad (1)$$

where g is a spanning subgraph of y, which can be a single arc or two adjacent arcs. In this article we assume that the dependency tree is a spanning projective tree. The model scores each subgraph using a linear feature vector model representation. Then scoring function $score(x, g)$ is,

$$score(x, g) = \mathbf{f}(x, g) \cdot \mathbf{w} \quad (2)$$

where $\mathbf{f}(x, g)$ is a high-dimensional feature vector based on features defined over g and x, and $\mathbf{w}$ refers to the weights for the features.

The maximum spanning tree is the highest scoring tree in $Y(G_x)$. The task of the decoding algorithms for an input sentence x is to find y^*, where

$$\begin{aligned} y^* &= \arg\max_{y \in Y(G_x)} \sum_{g \in y} score(x, g) \\ &= \arg\max_{y \in Y(G_x)} \sum_{g \in y} \mathbf{f}(x, g) \cdot \mathbf{w} \end{aligned} \quad (3)$$

C. Baseline parser

In our system, we use the decoding algorithm proposed by [2] and learn feature weights $\mathbf{w}$ using the Margin Infused Relaxed Algorithm (MIRA) [22], [19]. The decoding algorithm is an extension of the parsing algorithm of [23], which was a modified version of the CKY chart parsing algorithm. The algorithm independently parses the left and right dependents of a word and combines them later. There are two types of chart items [24]: 1) a *complete item* in which the words are unable to accept more dependents in a certain direction; and 2) an *incomplete item* in which the words can accept more dependents in a certain direction. In the algorithm, we create both types of chart items with two directions for all the word pairs in a given sentence. The direction of a dependency is from the head to the dependent. The right (left) direction indicates the dependent is on the right (left) side of the head. Larger chart items are created from pairs of smaller ones in a bottom-up style.

For graph-based parsing models, previous studies have defined different sets of features, including the first-order features, the second-order parent-siblings features, and the second-order parent-child-grandchild features [19], [24], [2]. Figure 2 shows the relations of tokens in dependency structures, where h and d refer to the head, the dependent, respectively, c refers to d's sibling or child, the structure of h and d is first-order, the one of h, c_h, and d is second-order parent-siblings structure, and the one of h, d, and c_{di} (or c_{do})

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

First-order
$[wp]_h$, $[wp]_d$, d(h, d)
$[wp]_h$, d(h, d)
w_d, p_d, d(h, d)
$[wp]_d$, d(h, d)
w_h, p_h, w_d, p_d, d(h, d)
p_h, w_h, p_d, d(h, d)
w_h, w_d, p_d, d(h, d)
w_h, p_h, $[wp]_d$, d(h, d)
p_h, p_b, p_d, d(h, d)
p_h, p_{h+1}, p_{d-1}, p_d, d(h, d)
p_{h-1}, p_h, p_{d-1}, p_d, d(h, d)
p_h, p_{h+1}, p_d, p_{d+1}, d(h, d)
p_{h-1}, p_h, p_d, p_{d+1}, d(h, d)
Second-order
p_h, p_d, p_c, d(h, d, c)
w_h, w_d, c$_w$, d(h, d, c)
p_h, $[wp]_c$, d(h, d, c)
p_d, $[wp]_c$, d(h, d, c)

Second-order (continue)
w_h, $[wp]_c$, d(h, d, c)
w_d, $[wp]_c$, d(h, d, c)
$[wp]_h$, $[wp]_{h+1}$, $[wp]_c$, d(h, d, c)
$[wp]_{h-1}$, $[wp]_h$, $[wp]_c$, d(h, d, c)
$[wp]_h$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_h$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)
$[wp]_{h-1}$, $[wp]_h$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_h$, $[wp]_{h+1}$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_{h-1}$, $[wp]_h$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)
$[wp]_h$, $[wp]_{h+1}$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)
$[wp]_d$, $[wp]_{d+1}$, $[wp]_c$, d(h, d, c)
$[wp]_{d-1}$, $[wp]_d$, $[wp]_c$, d(h, d, c)
$[wp]_d$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_d$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)
$[wp]_d$, $[wp]_{d+1}$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_d$, $[wp]_{d+1}$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)
$[wp]_{d-1}$, $[wp]_d$, $[wp]_{c-1}$, $[wp]_c$, d(h, d, c)
$[wp]_{d-1}$, $[wp]_d$, $[wp]_c$, $[wp]_{c+1}$, d(h, d, c)

TABLE I
BASE FEATURE TEMPLATES.

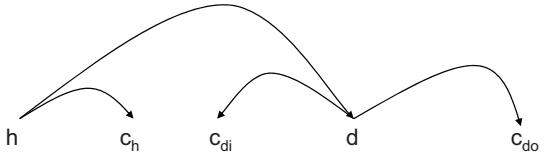

Fig. 2. Relations of tokens in dependency structures.

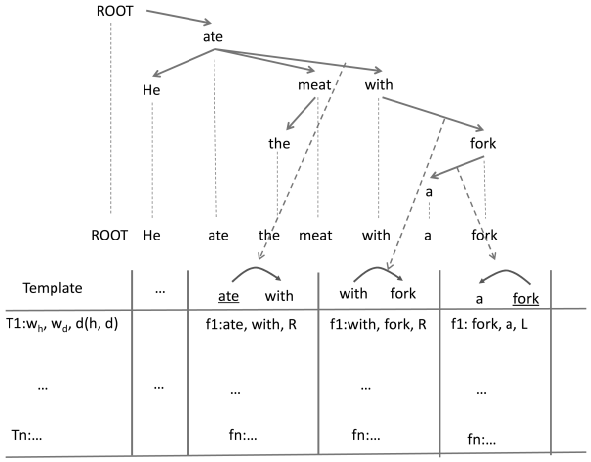

Fig. 3. Example of generating first-order features for dependency arcs.

is second-order parent-child-grandchild structure. [4] uses a richer set of features based on the above sets. We further extend the features by introducing more lexical features to the base features. The base feature templates are listed in Table I, where b refers to the word between h and d, $+1$ (-1) refers to the next (previous) word, w and p refer to the surface word and part-of-speech tag, respectively, $[wp]$ refers to the surface word or part-of-speech tag, $d(h, d)$ is the direction of the dependency relation between h and d, and $d(h, d, c)$ is the directions of the relation among h, d, and c.

We train a parser with the base features as the Baseline parser, and define $\mathbf{f}_b(x, g)$ as the base features and $\mathbf{w}_b$ as the corresponding weights. The scoring function becomes,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b \tag{4}$$

III. TWO MODELS TO INFER FEATURE EMBEDDINGS

Our goal is to learn a distributed representation for features, which is dense and low dimensional. We call the distributed feature representation feature embeddings. In the representation, each dimension represents a hidden-class of the features and is expected to capture a type of similarities or share properties among the features.

Our feature embeddings are inspired by word embeddings although there are several differences. Word embeddings can be induced using neural language models, which use neural networks as the underlying predictive model [25]. However, the training speed of neural language models is usually slow despite many approaches to improve it recently. In addition, a neural language model requires a segmental context, which is not avaiable for tree-structured features. [11] and [12] introduce the continuous Bag-of-Words (CBOW) and skip-gram models, which are efficient methods to directly learn high-quality word embeddings from large amounts of unstructured raw text. Since the two models do not involve dense matrix multiplications, the training speed is extremely fast.

We adapt the CBOW and skip-gram models for learning feature embeddings from large amounts of automatically parsed dependency tree data. Different from word embeddings, the input of our approach is features rather than words, and the feature representations are generated from tree structures instead of word sequences. Since the size of features is much larger than the vocabulary of words, feature embeddings result in a high computational cost. Thus in addition, we use the speed up techniques including subsampling of frequent features and Negative sampling in the learning stage [11].

A. Surrounding feature context

Given a sentence $x = w_1, w_2, ..., w_n$ and its corresponding dependency tree y, we can generate features based on the templates (defined in Table I). Figure 3 shows an example of generating first-order features for each dependency relation. We define the X-step context as a set of relations reachable

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

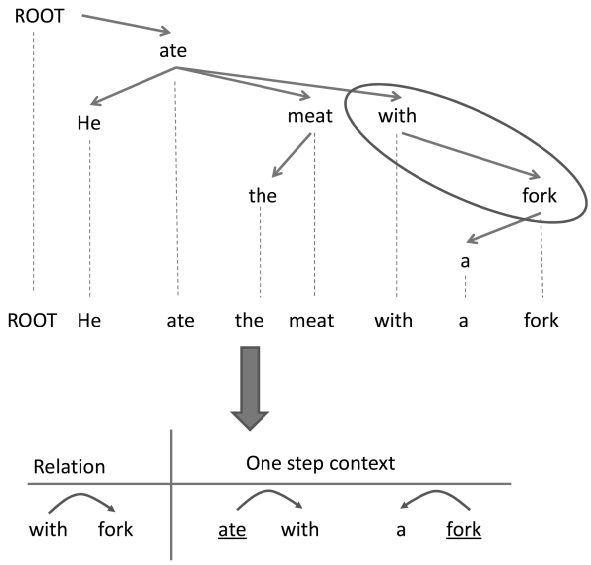

Fig. 4. An example of one-step context.

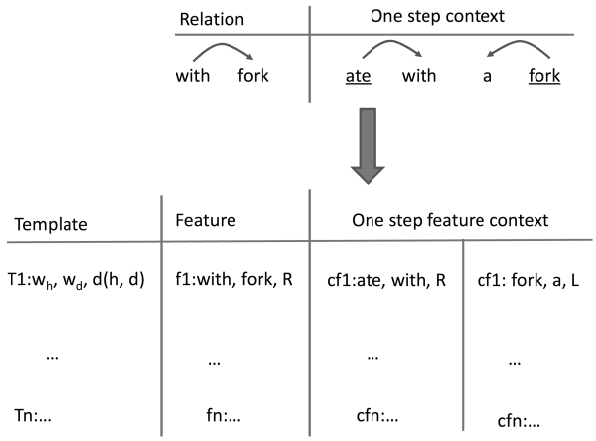

Fig. 5. One-step surrounding features.

within X steps from the current relation. Here one step refers to one dependency arc. For instance, the one-step context of "with $\curvearrowright$ fork" includes the surrounding relations that can be reached in one step such as "ate $\curvearrowright$ with" and "a $\curvearrowleft$ fork", as shown in Figure 4. In the figure, for the current relation between "with" and "fork", the relation between "ate" and "with" is in the one-step context, while the relation between "He" and "ate" is in the two-step context because it can be reached by two steps. A larger X results in more contextual features and thus might lead to a higher accuracy, but at the expense of training speed.

Based on the X-step context, we use surrounding features to represent the features on the current dependency relations. The surrounding features are defined on the relations in the X-step context. Take one-step context as an example. Figure 5 shows the representations for the current relation between "with" and "fork" in Figure 4. For the current relation and the relations in its one-step context, we generate the features based on the feature templates defined in Table I. In Figure 5 the current feature "f1:with, fork, R" can be represented by the surrounding features, "cf1:ate, with, R" and "cf1: fork, a, L" based on the template "T1:w_h, w_d, d(h, d)". Thus, all the features on the current relation are represented by the features on the relations in the one-step context. Note that to reduce computational cost, we generate every feature and its contextual features based on the same feature template. In the experiments, we use one-step context to represent the features, so that feature embeddings are defined over this context.

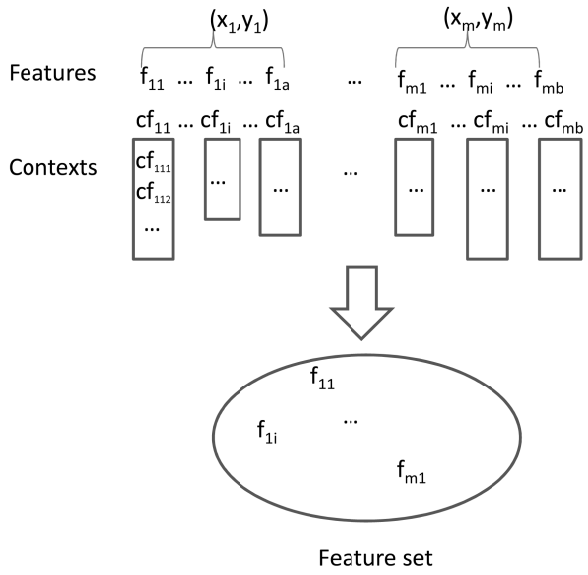

Fig. 6. Input feature set.

B. Two models

In this section, we adapt the models of [11] and [12] to infer feature embeddings. Based on the representation of surrounding context, the input of learning models is a set of features shown in Figure 6 and the output is feature embeddings. For each dependency tree in large amounts of auto-parsed data, we generate the base features that are associated with their surrounding contextual features. Then all the base features are put into a set. The learning models take the items one by one from the set during training.

1) Continuous Bag-of-Features (CBOF) Model: The idea of continuous Bag-of-Features model is inspired by the continuous Bag-of-Words model used in [12]. In our model, we use the associated surrounding features in the context to predict the current feature as shown in Figure 7. In the figure, we have a set of surrounding features $\{cf-1, cf-2, ..., cf-k\}$ to predict the current feature f. Similar to the CBOW model, we do not consider the order of features in the context in the prediction. In the CBOF model, we try to find feature representations that are useful for predicting the features on the current relation in a dependency tree. Given sentences and their corresponding dependency trees Y, the objective of the CBOF model is to maximize the log-likelihood over the data,

$$\sum_{y \in Y} \sum_{f \in F_y} \sum_{cf \in CF_f} \log(p(f|cf)) \quad (5)$$

where F_y is a set of features generated from tree y, CF_f is the set of surrounding features in the X-step context of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

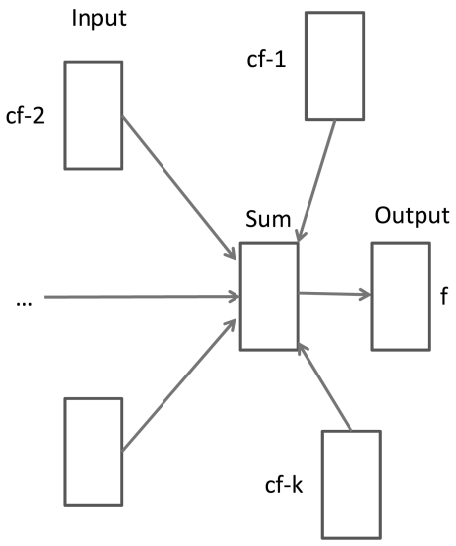

Fig. 7. The CBOF model.

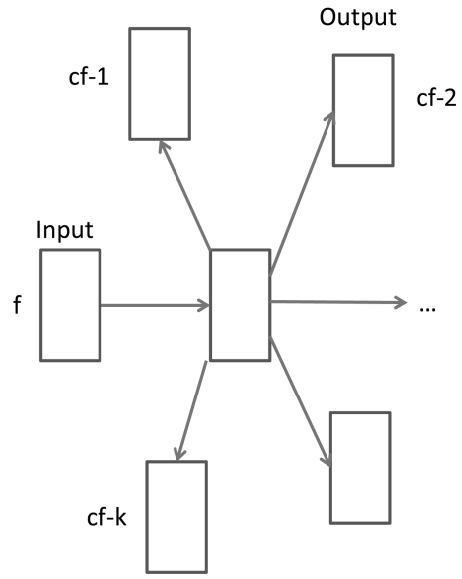

Fig. 8. The Skip-gram model.

feature f. $p(f|cf)$ can be computed by the softmax function [11] for which the input is cf and the output is f,

$$p(f|cf) = \frac{exp({v'_f}^{\mathrm{T}} v_{cf})}{\sum_{i=1}^{F} exp({v'_{f_i}}^{\mathrm{T}} v_{cf})} \quad (6)$$

where v_f and v'_f are the input and output vector representations of f, and F is the number of features in the feature table. The formulation is impractical for large data because the number of features is large (in the millions) and the computational cost for training the softmax structure is too high.

Several methods have been studied to make the training of embeddings feasible, including the hierarchical softmax variation [26], [27], [28], which reduces the computation cost, and the Negative sampling method, which is a simplified variation of Noise Contrastive Estimation[29], [30]. To compute the probabilities efficiently, we use the Negative sampling method proposed by [11], which approximates the probability by the correct example and K negative samples for each instance. The formulation to compute $log(p(f|cf))$ is,

$$\log \sigma({v'_f}^{\mathrm{T}} v_{cf}) + \sum_{k=1}^{K} \mathrm{E}_{f_k \sim P(f)} [\log \sigma(-{v'_{f_k}}^{\mathrm{T}} v_{cf})] \quad (7)$$

where $\sigma(z) = 1/(1 + exp(-z))$ and $P(f)$ is the noise distribution on the data. Following the setting of [11], we set K as 5 in our experiments.

We predict the features one by one in the set of features. Stochastic gradient ascent is used to perform the following iterative update after predicting the i^{th} feature,

$$\theta \leftarrow \theta + \alpha(\frac{\partial \sum_{cf} log(p(f_i|cf)}{\partial \theta}) \quad (8)$$

where α is the learning rate and θ includes the parameters of CBOW and the vector representations of features. The initial value of α is 0.025. If the log-likelihood does not improve significantly after one update, the rate is halved [31]. If the probability of the data does not improve again, the training stops.

2) Skip-gram model: The Skip-gram method models feature contexts in a different angle. In the Skip-gram model, we use the features on the current dependency arc to predict the surrounding features, as shown in Figure 8. In the figure, we use the current feature f to predict a set of surrounding features $\{cf-1, cf-2, ..., cf-k\}$. Given sentences and their corresponding dependency trees Y, the objective of the Skip-gram model is to maximize the log-likelihood,

$$\sum_{y \in Y} \sum_{f \in F_y} \sum_{cf \in CF_f} \log(p(cf|f)) \quad (9)$$

where F_y is a set of features generated from tree y and CF_f is the set of surrounding features in the X-step context of feature f. We also use the Negative sampling method to compute the log-likelihood, and the procedure of generating feature embeddings of the Skip-gram model is similar to the one of model CBOF.

The formulation to compute $log(p(cf|f))$ is,

$$\log \sigma({v'_{cf}}^{\mathrm{T}} v_f) + \sum_{k=1}^{K} \mathrm{E}_{cf_k \sim P(cf)} [\log \sigma(-{v'_{cf_k}}^{\mathrm{T}} v_f)] \quad (10)$$

where $\sigma(z) = 1/(1 + exp(-z))$ and $P(f)$ is the noise distribution on the data. Following the setting of [11], we set K to 5 in our experiments.

We also predict the set of features one by one. Stochastic gradient ascent is used to perform the following iterative update after predicting the i^{th} feature,

$$\theta \leftarrow \theta + \alpha(\frac{\partial \sum_{cf} log(p(cf_i|f)}{\partial \theta}) \quad (11)$$

where α is the learning rate and θ includes the parameters of the model and the vector representations of features. We compare the effectiveness of the two models in the experiments.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

$< j : T(f) \cdot \Phi(v_j) >$ for $j \in [1, d]$
$< j : T(f) \cdot \Phi(v_j), w_h >$ for $j \in [1, d]$

TABLE II
FE-BASED TEMPLATES.

C. Distributed representation

Based on the proposed surrounding context, we use the CBOF and Skip-gram models with the help of the Negative sampling method to learn feature embeddings. For each base template T_i, the distributed representations are stored in a matrix $\mathcal{M}_i \in \mathbb{R}^{d \times |\mathcal{F}_i|}$, where d is the number of dimensions (to be chosen in the experiments) and $|\mathcal{F}_i|$ is the size of the features $\mathcal{F}_i$ for T_i. For each feature $f \in \mathcal{F}_i$, its vector is $v_f = [v_1, ..., v_d]$.

IV. PARSING WITH FEATURE EMBEDDINGS

In this section, we discuss how to apply the feature embeddings to dependency parsing.

A. FE-based feature templates

The base parsing model contains only binary features, while the values in the feature embedding representation are real numbers that are not in a bounded range. If the range of the values is too large, they will exert too much more influence than the binary features. We confirm this in the preliminary experiments in which we used the continuous values directly, but obtained worse results. Thus, we define a function $\Phi(v_i)$ (in Section V) to convert the real values to discrete values. The vector $v_f = [v_1, ..., v_d]$ is converted into $v_f^{'} = [\Phi(v_1), ..., \Phi(v_d)]$.

We define a set of new templates for the parsing models, capturing feature embedding information, and being used in addition to the base features. Table II shows the new templates, where T(f) refers to the base template type of feature f. When generating the FE-based features, we do not generate any feature related to the surface form of the head if the word is not one of the Top-N most frequent words in the training data. This method can reduce the size of the feature sets and then speed up the system. After tuning on the development sets[2], we used Top-1000 for the experiments for this article.

B. FE parser

We combine the base features with the new features by a new scoring function,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b + \mathbf{f}_e(x, g) \cdot \mathbf{w}_e \qquad (12)$$

where $\mathbf{f}_b(x, g)$ refers to the base features, $\mathbf{f}_e(x, g)$ refers to the FE-based features, and $\mathbf{w}_b$ and $\mathbf{w}_e$ are their corresponding weights, respectively. The feature weights are learned during training using MIRA [22], [19].

We use the same decoding algorithm in the new parser as in the Baseline parser. The new parser is referred to as the FE Parser.

[2]The setting of Top-N only slightly affects the accuracy.

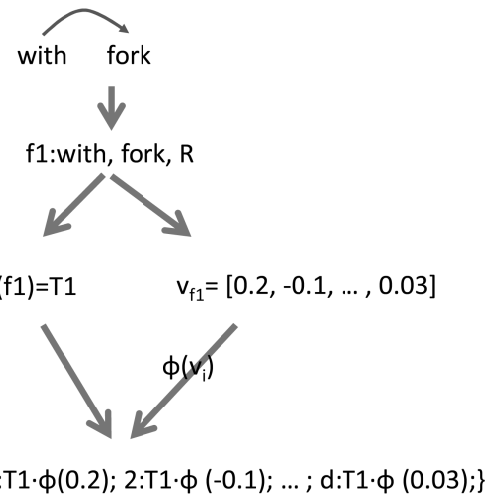

Fig. 9. An example of generating FE-based features.

V. IMPLEMENTATION DETAILS

A. Parsers

We implement the parsers based on the work of [2] with the base features defined in Table I. We train a second-order parser on the training data and use it to process the raw data.

B. Conversion functions

There are various functions to convert the real values in the vectors into discrete values. Here, we use a simple method. First, for the i^{th} base template, the values in the j^{th} dimension are sorted in decreasing order into the list L_{ij}. We divide the list into two halves for positive (L_{ij+}) and negative (L_{ij-}), respectively. We define two functions. In the first one, the function is defined as,

$$\Phi_1(v_j) = \begin{cases} +B1 & \text{if } v_j \text{ is in top 50\% in } L_{ij+} \\ +B2 & \text{if } v_j \text{ is in bottom 50\% in } L_{ij+} \\ -B1 & \text{if } v_j \text{ is in top 50\% in } L_{ij-} \\ -B2 & \text{if } v_j \text{ is in bottom 50\% in } L_{ij-} \end{cases}$$

In the second one, we define the function as,

$$\Phi_2(v_j) = \begin{cases} +B1 & \text{if } v_j \text{ is in top 50\% in } L_{ij+} \\ -B2 & \text{if } v_j \text{ is in bottom 50\% in } L_{ij-} \end{cases}$$

In Φ_2, we only consider the values ("+B1" and "-B2"), which have strong opinions (positive or negative) on dimensions and omit the values which are close to zero. We refer the systems with Φ_1 as M1 and the ones with Φ_2 as M2.

C. Generating FE-based features

We use an example to demonstrate how to generate new features based on the feature templates in practice. Suppose that we have a sentence "I ate the meat with a fork." and want to generate FE-based features for the relation between "with" and "fork", where "with" is the head and "fork" is the dependent. Figure 9 shows the example.

We demonstrate the generating procedure using the template T_1 = "w_h, w_d, d(h,d)" (the first base template in Table I),

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

	train	dev	test
PTB	2-21	22	23
CTB5	001-815 1001-1136	886-931 1148-1151	816-885 1137-1147

TABLE III
DATA SETS OF PTB AND CTB5.

name	numofwords	numofsents
BLLIP WSJ	43.4M	1.8M
Gigaword Xinhua	272.3M	11.7M

TABLE IV
INFORMATION OF RAW DATA.

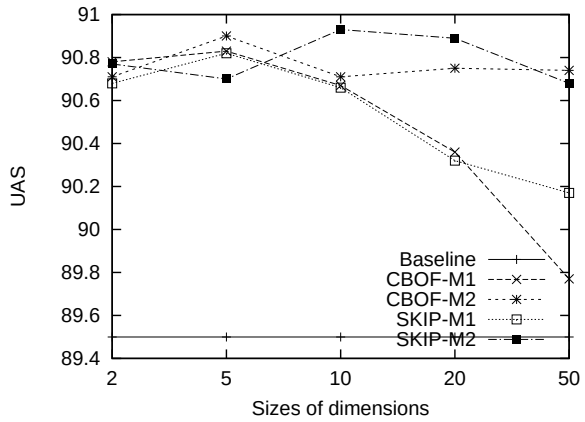

Fig. 10. Effect of different sizes of embeddings on the development data.

which contains the surface forms of the head, the dependent, and the direction of the dependency from h to d. We can have a base feature "with, fork, R", where "R" refers to the right arc direction. By looking up the matrix $\mathcal{M}_1$, we can get the embedding vector for the feature: $v_{f1} = [0.2, -0.1, ..., 0.03]$. According to Φ, we obtain a new vector $[\Phi(0.2), \Phi(-0.1), ..., \Phi(0.03)]$. Finally, we have a set of new features: $\{1 : T1 \cdot \Phi(0.2); 2 : T1 \cdot \Phi(-0.1); ...; d : T1 \cdot \Phi(0.03); \}$. In this way, we can generate all the new features for the graph-based model. The number of new features is relative small, totally $4 \times d \times |T|$ for M1 and $2 \times d \times |T|$ for M2, respectively, where T is the set of base templates. d and $|T|$ are often small (in the hundreds).

VI. EXPERIMENTS

We conducted experiments on the standard data sets of English and Chinese, respectively.

A. Data sets

We used the Penn Treebank (PTB) [16] to generate the English data sets and the Chinese Treebank version 5.1 (CTB5) [17] to generate the Chinese data sets. "Penn2Malt"[3] was used to convert the data into dependency structures with the English head rules of [32] and the Chinese head rules of [33]. The details of data splits are listed in Table III, where the data partition of Chinese were chosen to match previous work [34], [35], [36].

Following the work of [13], we used a tagger trained on training data to provide part-of-speech (POS) tags for the development and test sets, and used 10-way jackknifing to generate part-of-speech tags for the training set. For English we used the MXPOST [37] tagger and for Chinese we used a CRF-based tagger with the feature templates defined in [38]. We used gold-standard segmentation in the CTB5 experiments. The accuracies of part-of-speech tagging are 97.32% for English and 93.61% for Chinese on the test sets, respectively.

To obtain feature contexts, we processed raw data to get dependency trees. For English, we used the BLLIP WSJ Corpus Release 1 [39].[4] For Chinese, we used the Xinhua portion of Chinese Gigaword[5] Version 2.0 (LDC2009T14) [40]. The statistical information of raw data sets is listed in Table IV. The MXPOST part-of-speech tagger and the Baseline dependency parser trained on the training data were used to process the sentences of the BLLIP WSJ corpus. For Chinese, we need to perform word segmentation and part-of-speech tagging before parsing. The MMA system [41] trained on the training data was used to perform word segmentation and tagging, and the Baseline parser was used to parse the sentences in the Gigaword corpus.

We report the parser quality by the unlabeled attachment score (UAS), i.e. the percentage of tokens (excluding all punctuation tokens) with the correct HEAD. We also report the scores on complete dependency trees evaluation (COMP).

B. Development experiments

In this section, we use the development data set of English to investigate the effect of different vector sizes of feature embeddings and compare the systems with M1 and M2 (defined in Section V-B). To reduce the training time, we used 10% of labeled training data to train the parsing models.

[6] reported that the optimal size of word embedding dimensions was task-specific for NLP tasks. Here, we also investigated the effect of different sizes of embedding dimensions on dependency parsing. Figure 10 shows the effect on UAS scores as we varied the vector sizes, where CBOF-M1/M2 refers to the system with CBOF model and M1/M2, SKIP-M1/M2 refer to the system with Skip-gram model and M1/M2 respectively. The systems with FE-based features always outperformed the Baseline. The curves of the parsers with M2 were almost flat and we found that the parsers with M1 performed worse as the sizes increased. Overall, the systems with M2 performed better than the ones with M1. For SKIP-M2, 10-dimensional embeddings achieved the highest score among all the systems. For CBOF-M2, 5-dimensional embeddings performed the best among the CBOF-based systems.

Based on the above observations, we chose two systems for further evaluations: 5-dimensional embeddings for CBOF-M2 and 10-dimensional embeddings for SKIP-M2.

[3]http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html
[4]We excluded the texts of PTB from the BLLIP WSJ Corpus.
[5]We excluded the texts of CTB5 from the Gigaword data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

	UAS	COMP
Baseline	92.78	48.08
CLU	93.37	49.26
SKIP-M2	93.74	50.82
CBOF-M2	93.62	50.08
Koo2010	93.04	N/A
Zhang2011	92.9	48.0
Koo2008	93.16	N/A
Suzuki2009	93.79	N/A
Chen2009	93.16	47.15
Zhou2011	92.64	46.61
Suzuki2011	94.22	N/A
Chen2013	93.77	51.36

TABLE V
RESULTS ON ENGLISH DATA. N/A=NOT AVAILABLE.

C. Main results on English data

We trained CBOF-M2 and SKIP-M2 on the full training data and evaluated them on the testing data for English. The results are shown in Table V. The parsers using the FE-based features consistently outperformed the Baseline. For SKIP-M2 and CBOF-M2, we obtained absolute improvements of 0.96 and 0.84 UAS points, respectively. As for the COMP scores, SKIP-M2 achieved absolute improvement of 2.74 over the Baseline. The improvements were significant by McNemar's Test ($p < 10^{-7}$) [42]. We also added the cluster-based features of [13] to our baseline system listed as "CLU" in Table V. The parsers using the FE-based features outperform CLU too.

We listed the performance of the related systems in Table V, where Koo2010 refers to the supervised system of [3], which is based on a third-order graph-based model, Zhang2011 refers to the supervised system of [5], which uses rich non-local features in a transition-based model, Koo2008 refers to the semi-supervised system of [13], which uses a second-order graph-based model together with Brown word-cluster based features, Suzuki2009 refers to the semi-supervised system of [43], which uses a semi-supervised structured conditional model[44], Chen2009 refers to a semi-supervised system [45] that learns frequency-based subtree features from auto-parsed data, Zhou2011 refers to the semi-supervised system of [46], which exploits web-derived selectional preference, Suzuki2011 refers to the semi-supervised system of [47], which uses a condensed feature representation, and Chen2013 refers to the semi-supervised system of [48], which uses frequency-based meta-features learned from auto-parsed data.

From the table, we found that our FE parsers obtained the comparable accuracy with the previous state-of-the-art systems. Suzuki2011 [47] reported the best reported result by combining their method with the method of Koo et al.[13]. We believe that the performance of our parser can be further enhanced by integrating their methods.

D. Main results on Chinese data

We also evaluated the systems on the testing data for Chinese. The results are shown in Table VI. We also added the cluster-based features of [13] to our baseline system listed as "CLU" in Table VI. Similar to the results on English, the parsers using the FE-based features consistently outperformed the Baselines.

	POS	UAS	COMP
Baseline	93.61	81.04	29.73
CLU	93.61	81.52	30.23
CBOF-M2	93.61	82.78	30.86
SKIP-M2	93.61	82.94	31.72
Li2011	93.08	80.74	29.11
Hatori2011	93.94	81.33	29.90
Li2012	94.51	81.21	N/A
Chen2013	N/A	83.08	32.21

TABLE VI
RESULTS ON CHINESE DATA.

We listed the performance of the related systems[6] on Chinese in Table VI, where Li2011 refers to the system of [35], Hatori2011 refers to the system of [36], Li2012 refers to the unlabeled parser of [50], and Chen2013 refers to the system of [48]. From the table, we found that the scores of our FE parsers were higher than most of the related systems and comparable with the results of Chen2013, which was the best reported scores so far. We believe that the performance of our parser can be further enhanced by enlarging X-step contexts and learn better feature embeddings.

VII. RELATED WORK

Learning feature embeddings are related to two lines of research: deep learning models for NLP, and semi-supervised dependency parsing.

Recent studies used deep learning models in a variety of NLP tasks. [6] applied word embeddings to chunking and Named Entity Recognition (NER). [9] designed a unified neural network to learn distributed representations that were useful for part-of-speech tagging, chunking, NER, and semantic role labeling. They tried to avoid task-specific feature engineering. [51] proposed a Compositional Vector Grammar, which combined PCFGs with distributed word representations. [52] investigated Chinese character embeddings for Chinese word segmentation and part-of-speech tagging. [49] directly applied word embeddings to Chinese dependency parsing. [53] inferred word embeddings based on contexts extracted from dependency trees. In most cases, words or characters were the inputs to the learning systems and they applied word/character embeddings to their tasks. Our work is different from theirs in that we explore distributed representations at the feature level and we can make full use of well-established hand-designed features.

In our work, we use large amounts of raw data to infer feature embeddings. There are several previous studies relevant to using raw data on dependency parsing. [13] used the Brown algorithm to learn word clusters from a large amount of unannotated data and defined a set of word cluster-based features for dependency parsing models. [43] adapted a Semi-supervised Structured Conditional Model (SS-SCM) [44] to dependency parsing. [47] reported the best results so far on the standard test sets of PTB using a condensed feature representation combined with the word cluster-based features

[6]We did not include the result (83.96) of [49] because their part-of-speech tagging accuracy is 97.7%, much higher than ours and other work. Their tagger includes rich external resources.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2014.2365359, IEEE/ACM Transactions on Audio, Speech, and Language Processing

of [13]. [48] mapped the base features into predefined types using the information of frequencies counted in large amounts of auto-parsed data. The work of [47] and [48] were to perform feature clustering. [54] presented a semi-supervised learning algorithm named alternating structure optimization for text chunking. They used a large projection matrix to map sparse base features into a small number of high level features over a large number of auxiliary problems. One of the advantages of our approach is that it is simpler and more general than that of [54]. Our approach can easily be applied to other tasks by defining new feature contexts.

VIII. Conclusion

In this article, we have presented an approach to learning feature embeddings for dependency parsing from large amounts of raw data. The raw sentences were first parsed by a baseline system and then we obtained an auto-parsed data. Each model feature was represented by the surrounding features on the dependency trees. Based on the representation of surrounding context, we proposed two learning methods to infer feature embeddings. Finally, we represented a set of new features based on the learned feature embeddings, which was used with the base features in a graph-based model. When tested on both English and Chinese, our method significantly improved the performance over strong baselines and provided comparable accuracies with the best systems in the literature.

For future work, there are several ways in which this research could be extended. First, we plan to use a larger data to infer the feature embeddings. Second, we could apply the proposed approach to other languages (for example, Japanese). Third, we could extend to labeled parsing instead of unlabeled parsing. Finally, we could extend the approach to the constituency parsing task.

Acknowledgments

Wenliang Chen and Min Zhang were supported by the National Natural Science Foundation of China (Grant No. 61203314 and 61373095) and Yue Zhang was supported by MOE grant 2012-T2-2-163. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

[1] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret, "The CoNLL 2007 shared task on dependency parsing," in *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, 2007, pp. 915–932.

[2] X. Carreras, "Experiments with a higher-order projective dependency parser," in *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 957–961.

[3] T. Koo and M. Collins, "Efficient third-order dependency parsers," in *Proceedings of ACL 2010*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 1–11.

[4] B. Bohnet, "Top accuracy and fast dependency parsing is not a contradiction," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, August 2010, pp. 89–97. [Online]. Available: http://www.aclweb.org/anthology/C10-1011

[5] Y. Zhang and J. Nivre, "Transition-based dependency parsing with rich non-local features," in *Proceedings of ACL-HLT2011*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 188–193. [Online]. Available: http://www.aclweb.org/anthology/P11-2033

[6] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of ACL 2010*. Association for Computational Linguistics, 2010, pp. 384–394.

[7] D. Lin, "Using syntactic dependency as local context to resolve word sense ambiguity," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*. Madrid, Spain: Association for Computational Linguistics, July 1997, pp. 64–71. [Online]. Available: http://www.aclweb.org/anthology/P97-1009

[8] J. Curran, "Supersense tagging of unknown nouns using semantic similarity," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 26–33. [Online]. Available: http://www.aclweb.org/anthology/P05-1004

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[10] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119. [Online]. Available: http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips26/1421.pdf

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[13] T. Koo, X. Carreras, and M. Collins, "Simple semi-supervised dependency parsing," in *Proceedings of ACL-08: HLT*, Columbus, Ohio, June 2008.

[14] J. R. Firth, *A synopsis of linguistic theory, 1930-1955*, 1957.

[15] R. McDonald and J. Nivre, "Characterizing the errors of data-driven dependency parsing models," in *Proceedings of EMNLP-CoNLL*, 2007, pp. 122–131.

[16] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguisticss*, vol. 19, no. 2, pp. 313–330, 1993.

[17] N. Xue, F. Xia, F. dong Chiou, and M. Palmer, "Building a Large Annotated Chinese Corpus: the Penn Chinese Treebank," *Journal of Natural Language Engineering*, vol. 11, no. 2, pp. 207–238, 2005.

[18] W. Chen, Y. Zhang, and M. Zhang, "Feature embeddings for dependency parsing," in *Proceedings of Coling 2014*, August 2014.

[19] R. McDonald, K. Crammer, and F. Pereira, "Online large-margin training of dependency parsers," in *Proceedings of ACL 2005*. Association for Computational Linguistics, 2005, pp. 91–98.

[20] J. Nivre and R. McDonald, "Integrating graph-based and transition-based dependency parsers," in *Proceedings of ACL-08: HLT*, Columbus, Ohio, June 2008.

[21] B. Bohnet and J. Nivre, "A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing," in *Proceedings of EMNLP-CoNLL 2012*. Association for Computational Linguistics, 2012, pp. 1455–1465.

[22] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *J. Mach. Learn. Res.*, vol. 3, pp. 951–991, 2003.

[23] J. Eisner, "Three new probabilistic models for dependency parsing: An exploration," in *Proceedings of COLING1996*, 1996, pp. 340–345.

[24] R. McDonald and F. Pereira, "Online learning of approximate dependency parsing algorithms," in *Proceedings of EACL 2006*, 2006, pp. 81–88.

[25] Y. Bengio, "Neural net language models," in *Scholarpedia*, 2008, p. 3881.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of ICLR Workshop*, 2013.

[27] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of AISTATS*, 2005, pp. 246–252.

[28] A. Mnih and G. Hinton, "A scalable hierarchical distributed langauge model," in *Proceedings of NIPS*, 2009, pp. 1081–1088.

[29] M. Gutmann and A. Hyvarinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistica," *The Journal of Machine Learning Research*, vol. 13, pp. 307–361, 2012.

[30] A. Mhih and H. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of ICML*, 2012.

[31] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, and J. Cernocky, "Neural network based language models for highly inflective languages," in *Proceedings of ICASSP 2009*. IEEE, 2009, pp. 4725–4728.

[32] H. Yamada and Y. Matsumoto, "Statistical dependency analysis with support vector machines," in *Proceedings of IWPT 2003*, 2003, pp. 195–206.

[33] Y. Zhang and S. Clark, "A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing," in *Proceedings of EMNLP 2008*, Honolulu, Hawaii, October 2008, pp. 562–571.

[34] X. Duan, J. Zhao, and B. Xu, "Probabilistic models for action-based chinese dependency parsing," in *Proceedings of ECML/ECPPKDD*, Warsaw, Poland, 2007.

[35] Z. Li, M. Zhang, W. Che, T. Liu, W. Chen, and H. Li, "Joint Models for Chinese POS Tagging and Dependency Parsing," in *Proceedings of EMNLP 2011*, UK, July 2011.

[36] J. Hatori, T. Matsuzaki, Y. Miyao, and J. Tsujii, "Incremental Joint POS Tagging and Dependency Parsing in Chinese," in *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, November 2011, pp. 1216–1224. [Online]. Available: http://www.aclweb.org/anthology/I11-1136

[37] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proceedings of EMNLP 1996*, 1996, pp. 133–142.

[38] Y. Zhang and S. Clark, "Joint word segmentation and POS tagging using a single perceptron," in *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 888–896. [Online]. Available: http://www.aclweb.org/anthology/P/P08/P08-1101

[39] E. Charniak, D. Blaheta, N. Ge, K. Hall, J. Hale, and M. Johnson, "BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43," *Linguistic Data Consortium*, 2000.

[40] C.-R. Huang, "Tagged Chinese Gigaword Version 2.0, LDC2009T14," *Linguistic Data Consortium*, 2009.

[41] C. Kruengkrai, K. Uchimoto, J. Kazama, Y. Wang, K. Torisawa, and H. Isahara, "An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging," in *Proceedings of ACL-IJCNLP2009*. Suntec, Singapore: Association for Computational Linguistics, August 2009, pp. 513–521.

[42] J. Nivre, J. Hall, and J. Nilsson, "Memory-based dependency parsing," in *Proc. of CoNLL 2004*, 2004, pp. 49–56.

[43] J. Suzuki, H. Isozaki, X. Carreras, and M. Collins, "An empirical study of semi-supervised structured conditional models for dependency parsing," in *Proceedings of EMNLP2009*. Singapore: Association for Computational Linguistics, August 2009, pp. 551–560.

[44] J. Suzuki and H. Isozaki, "Semi-supervised sequential labeling and segmentation using Giga-word scale unlabeled data," in *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 665–673.

[45] W. Chen, J. Kazama, K. Uchimoto, and K. Torisawa, "Improving dependency parsing with subtrees from auto-parsed data," in *Proceedings of EMNLP 2009*, Singapore, August 2009, pp. 570–579.

[46] G. Zhou, J. Zhao, K. Liu, and L. Cai, "Exploiting web-derived selectional preference to improve statistical dependency parsing," in *Proceedings of ACL-HLT2011*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 1556–1565. [Online]. Available: http://www.aclweb.org/anthology/P11-1156

[47] J. Suzuki, H. Isozaki, and M. Nagata, "Learning condensed feature representations from large unsupervised data sets for supervised learning," in *Proceedings of ACL2011*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 636–641. [Online]. Available: http://www.aclweb.org/anthology/P11-2112

[48] W. Chen, M. Zhang, and Y. Zhang, "Semi-supervised feature transformation for dependency parsing," in *Proceedings of EMNLP 2013*. Seattle, Washington, USA: Association for Computational Linguistics, October 2013, pp. 1303–1313. [Online]. Available: http://www.aclweb.org/anthology/D13-1129

[49] X. Wu, J. Zhou, Y. Sun, Z. Liu, D. Yu, H. Wu, and H. Wang, "Generalization of words for chinese dependency parsing," in *Proceedings of IWPT 2013*, 2013, pp. 73–81.

[50] Z. Li, M. Zhang, W. Che, and T. Liu, "A Separately Passive-Aggressive Training Algorithm for Joint POS Tagging and Dependency Parsing," in *Proceedings of the 24rd International Conference on Computational Linguistics (Coling 2012)*. Mumbai, India: Coling 2012 Organizing Committee, 2012.

[51] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of ACL 2013*. Citeseer, 2013.

[52] X. Zheng, H. Chen, and T. Xu, "Deep Learning for Chinese Word Segmentation and POS Tagging," in *Proceedings of EMNLP 2013*. Association for Computational Linguistics, 2013, pp. 647–657.

[53] O. Levy and Y. Goldberg, "Dependency Based Word Embeddings," in *Proceedings of ACL 2014(short)*. Association for Computational Linguistics, 2014.

[54] R. Ando and T. Zhang, "A high-performance semi-supervised learning method for text chunking," *ACL*, 2005.