Expert Systems with Applications 41 (2014) 1716–1722

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# ReliAble dependency arc recognition

Wanxiang Che, Jiang Guo, Ting Liu *

*School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China*

ARTICLE INFO

*Keywords:*
Natural language processing
Syntactic parsing
Dependency parsing
RADAR
Binary classification

ABSTRACT

We propose a novel natural language processing task, ReliAble dependency arc recognition (RADAR), which helps high-level applications better utilize the dependency parse trees. We model RADAR as a binary classification problem with imbalanced data, which classifies each dependency parsing arc as correct or incorrect. A logistic regression classifier with appropriate features is trained to recognize reliable dependency arcs (correct with high precision). Experimental results show that the classification method can outperform a probabilistic baseline method, which is calculated by the original graph-based dependency parser.



## 1. Introduction

As a fundamental task of natural language processing, dependency parsing has become increasingly popular in recent years. It aims to find a dependency parse tree among words for a sentence. Fig. 1 shows an example of dependency parse tree for a sentence, where `sbj` is a subject, `obj` is an object, etc. (Johansson & Nugues, 2007). Dependency parsing are widely used: in biomedical text mining (Kim, Ohta, Pyysalo, Kano, & Tsujii, 2009), as well as in textual entailment (Androutsopoulos & Malakasiotis, 2010), information extraction (Wu & Weld, 2010; Banko, Cafarella, Soderland, Broadhead, & Etzioni, 2007) and sentiment analysis (Meena & Prabhakar, 2007).

The performance of dependency parsing has increased recently (Kübler, McDonald, & Nivre, 2009). However, when we migrate dependency parsing systems from laboratory demonstrations to high-level applications, even the best parser available today still encounter some serious difficulties.

First of all, parsing performance usually dramatically degrades in real fields because of domain migration. Secondly, since every parser inevitably will make some mistakes during decoding, outputs from any dependency parser are always fraught with a variety of errors. Thus, in some high-level applications which expect to use correct parsing results, it is extremely important to be able to predict the reliability of the auto-parsed results. If these applications just use correct parsing results and ignore incorrect results, their performances may be improved further. For instance, if an entity relation extraction (a kind of information extraction) system, which depends on parsing results heavily (Zhang, Zhang, Su, & Zhou, 2006), only extracts relations from correct parsing sentences, then the system can extract more accurate relations and import less wrong relations through incorrect parsing results. Although some implied relations in those incorrect parsing sentences are missed, these missing relations may be extracted from other sentences that can be parsed correctly while zooming in the data to the whole Web.

Most large-margin based training algorithm for dependency parsing output models that predict a single parse tree of the input sentence, with no additional confidence information about the correctness of it. Therefore, an interesting problem is how to judge a parsing result as correct or not. However, it is difficult to obtain a parse tree in which all sub-structures are parsed correctly. CoNLL 2009 Shared Task results show that only about 40% English and 35% Chinese sentences can be parsed complete correctly (Hajič et al., 2009b). Some previous studies have addressed the problem to recognize reliable parsing results (Reichart & Rappoport, 2007; Dell'Orletta & Venturi, 2011; Kawahara & Kurohashi, 2010; Ravi, Knight, & Soricut, 2008). A parsing result is reliable when, the result is correct with high precision. However, all these studies focus on judging if the parsing results of a whole sentence are reliable or not, which can cause the following problems:

1. The reliable parsing results may still include some wrong parsing sub-structures. Different applications need different key sub-structures, such as backbone structures that are keys for semantic role labeling (Gildea & Jurafsky, 2002) and branch structures are important for multiword expression (Sag, Baldwin, Bond, Copestake, & Flickinger, 2002). If these key sub-structures are parsed incorrectly, even though the whole sentence is parsed with a high reliability, the tiny errors will be still harmful to these given applications. This problem results in a low precision.

* Corresponding author. Tel.: +86 13936137628.
*E-mail address:* tliu@ir.hit.edu.cn (T. Liu).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.070

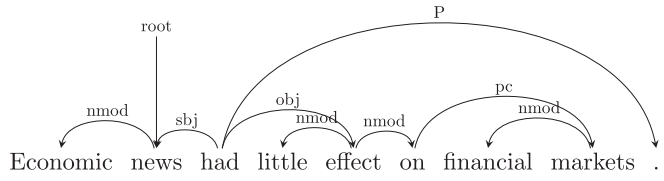


**Fig. 1.** An example of dependency parse tree.

2. The unreliable parsing results may include some useful sub-structures but should not be discarded totally. For instance, extracting entity relations is possible if the parse tree path is correct between two entities despite other parts of the sentence being incorrectly parsed. Discarding unreliable parsing sentences can result in a low recall.

Therefore, dependency arcs, novel reliability measuring objects for dependency parsing are proposed. A reliable dependency happens when a word can find its parent and label the dependency relation between them correctly with high precision. Once all reliable dependency arcs in a sentence are found, the corresponding parse paths or sub-trees from them can be mapped out. These reliable sub-structures can be used according to the needs of different applications. In paying attention to reliable parts and ignoring unreliable ones in a sentence, the precision of applications can be improved. Meanwhile, when the number of reliable sub-structures is more than that extracted from reliable whole sentences, higher recall can be obtained.

The problem of ReliAble Dependency Arc Recognition (RADAR) can be regarded a binary classification problem. The positive examples are the correctly predicted arcs and the others are the negative examples. Thus, the problem can be converted to find appropriate classifiers and proper features. Different from normal binary classification problems, the data are not balanced for RADAR. For the state-of-the-art dependency parser, the LAS (Labeled Attachment Score) can achieve about 80% in Chinese data set and 90% in English data set (Hajič et al., 2009b), which means that the ratio of the number of correct dependency arcs to the number of incorrect dependency arcs is 4:1 for Chinese and 9:1 for English. Aside from learning from the imbalanced data, how to evaluate RADAR is another issue. The normal evaluation methods based on accuracy are not suitable for the problem. If an incorrect dependency arc is recognized as a correct arc, the cost is larger than the opposite scenario. In addition, the classification accuracy would not be a suitable evaluation metric in an imbalanced scenario. Therefore, there is a need to find more appropriate evaluation criteria.

The rest of the present paper is organized as follows. Section 2 presents related work. Section 3 describes the proposed method. Section 4 discusses the present experimental setting and results. We conclude and set the direction of the future work in Sections 5 and 6 respectively.

## 2. Related work

To the best of our knowledge, Yates, Schoenmackers, and Etzioni (2006) was the first work to address explicitly the parsing reliability recognition problem. They detected erroneous parses using web-based semantics. In addition, an ensemble method based on different parsers trained on different data sampled from a training corpus to select high quality parsing results was proposed as well (Reichart & Rappoport, 2007). Dell'Orletta and Venturi (2011) was another study that detect reliable dependency parses with some heuristic features. Kawahara and Uchimoto (2008) classified sentences into two classes, reliable and unreliable, with a binary classifier. Ravi et al. (2008) predicted the accuracy of a parser on sets of sentence by fitting a real accuracy curve with linear regression algorithm.[1] However, all these works focused on recognizing reliable parsing results of whole sentences and caused corresponding problems for some applications as discussed in Section 1.

Although the parsing reliability recognition of whole sentences can be used in Active Learning (Settles, 2010) or Semi/Un-supervised Learning (Goldwasser, Reichart, Clarke, & Roth, 2011), recognizing sub-structures reliability is also useful. For instance, some studies (van Noord, 2007; Chen, Kawahara, Uchimoto, & Zhang, 2008, 2009) used sub-trees or word pairs extracted from a large auto-parsed corpus to help the dependency parser. However, the confidence of a sub-tree or a word pair is only expressed by its count that appears in the corpus. Therefore, their methods may be biased toward frequently appearing sub-trees or word pairs, which may be incorrect, and penalizes the sparse but correct ones.

The studies most relevant to ours are done by Atserias, Attardi, Simi, and Zaragoza (2010) and Avihai Mejer (2012). They both reported the similar problem with ours. Atserias et al. (2010) shows how to use the probability scores that a transition-based parser normally computes, in order to assigning a confidence score to parse trees. They assign such score to each arc and the active learning application uses the worst. Another independent work done by Avihai Mejer (2012) describes several methods for estimating the confidence in the per-edge correctness of a predicted dependency parse. The best method they confirmed in their study is based on model re-sampling, which is inefficient. Our work differs in that we proposed a novel supervised approach which makes use of additional information as the features for learning models.

## 3. Method description

This section introduces the dependency parsing model and a method to estimate the probability of each dependency arc. A binary classification method to recognize reliable arcs follows. Besides a classifier, the classification method includes three sorts of features and a process to construct training data.

### 3.1. Graph-based dependency parsing

Given an input sentence $\mathbf{x} = w_1 \dots w_n$, a dependency tree is denoted by $\mathbf{d} = \{(h,m,l) : 0 \leqslant h \leqslant n, 0 < m \leqslant n, l \in \mathcal{L}\}$, where $(h,m,l)$ represents a dependency arc $w_h \rightarrow w_m$ whose *head* word (or *father*) is $w_h$ and *modifier* (or *child*) is $w_m$ with a dependency label $l$, and $\mathcal{L}$ is the set of all possible dependency relation labels. The artificial node $w_0$, which always points to the root of the sentence, is used to simplify the formalizations.

Then, an optimal dependency tree $\hat{\mathbf{d}}$ is determined based on $\mathbf{x}$:

$$\hat{\mathbf{d}} = \arg\max_{\mathbf{d}} Score(\mathbf{x}, \mathbf{d})$$

Recently, graph-based dependency parsing has gained interest due to its state-of-the-art performance (Kübler et al., 2009). Graph-based dependency parsing views the problem as finding the highest scoring tree from a directed graph. Based on dynamic programming decoding, it can find efficiently an optimal tree in a huge search space. In a graph-based model, the score of a dependency tree is factored into scores of small parts (sub-trees):

$$Score(\mathbf{x}, \mathbf{d}) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{d}) = \sum_{p \subseteq \mathbf{d}} Score(\mathbf{x}, p)$$

where $\mathbf{f}(\mathbf{x},\mathbf{d})$ refers to the feature vector and $\mathbf{w}$ is the corresponding weight vector, $p$ is a scoring part that contains one or more dependence arcs in the dependency tree $\mathbf{d}$.

[1] When the size of a set is 1, the accuracy of a sentence can be predicted.

The present paper devotes to realize RADAR in graph-based parsing algorithm. The most intuitive method is to use the probability of a dependency arc to denote its reliability. However, the graph-based parsing method is based on a discriminative model but not a generative model, therefore, we only can obtain the score of each arc ($Score(\mathbf{x},p)$) but not its accurate probability. To overcome this difficulty, a study of Koo, Globerson, Carreras, and Collins (2007)'s idea is used to estimate the probability of the dependency arc. Before estimating the arc probability, the probability of a dependency tree $\mathbf{d}$, which can be obtained by exponentiating and renormalizing the score $Score(\mathbf{x},\mathbf{d})$, has to be computed:

$$P(\mathbf{d}|\mathbf{x};M) = \frac{\exp(Score(\mathbf{x},\mathbf{d}))}{Z(\mathbf{x};M)}$$

$$Z(\mathbf{x};M) = \sum_{\mathbf{d}'\in\mathcal{T}(\mathbf{x})} \exp(Score(\mathbf{x},\mathbf{d}'))$$

where $M$ is the dependence parsing model, and $Z(\mathbf{x};M)$ is a normalization factor. $\mathcal{T}(\mathbf{x})$ is the set of all possible parse trees for the sentence.

Given the conditional distribution $P(\mathbf{d}|\mathbf{x};M)$, the probability of a dependency arc $(h,m,l)$ is:

$$P((h,m,l)|\mathbf{x};M) = \sum_{\mathbf{d}'\in\mathcal{T}(x):(h,m,l)\in\mathbf{d}'} P(\mathbf{d}'|\mathbf{x},M)$$

Note that both probabilities require a summation over the set $\mathcal{T}(\mathbf{x})$, which is exponential in sentence size $n$. For the sake of simplicity, we used the $k$-best dependency tree list in place of $\mathcal{T}(\mathbf{x})$, where $k$ is set to 1000 in the experiments.

### 3.2. RADAR as binary classification

In a dependency parse tree, all dependency arcs are classified into two classes: positive (the modifier and head words of an arc are extracted correctly and the syntactic relation is correct.[2]) and negative (otherwise). RADAR can be regarded as a binary classification problem naturally. For the classification problem, three questions need to be answered: 1. Which classifier is used? 2. What features are extracted? and 3. How is the training data constructed?

#### 3.2.1. Classifier

The logistic regression classifier (Bishop, 2006) is used in the experiments. The main reason to select logistic regression is that it can estimate the probability of each class, which will be used as the criteria for RADAR later. In addition, logistic regression is fast for training and effective for predicting. The present study used L2-regularized logistic regression, which solves:

$$\arg\min_{w} \frac{\|w\|^2}{2} + C^+ \sum_{\{i|y_i=+1\}}^{n_+} log(1+e^{(-y_i w^T x_i)}) + C^- \sum_{\{i|y_i=-1\}}^{n_-} log(1+e^{(-y_i w^T x_i)})$$

where $(x_i,y_i)$ is the $i$th training sample and $w$ is the feature weight vector. $C^+$ and $C^-$ are the penalties of constraints violation for positive and negative classes respectively, which can be used to handle the imbalanced data problem. A larger $C^-$ can be set to prevent the classifier from tending to recognize incorrect arcs as correct ones (the majority class).

#### 3.2.2. Features

Besides a classifier algorithm, features are also important for a classification system. To predict whether a dependency arc is correct or not, we defined three sorts of features. Below is the detail description of the features and intuitions.

1. Text-based features relate only to the original text of a sentence.
   - The length of a sentence: Longer sentences are harder to be parsed correctly than shorter sentences. We group the lengths into three buckets: with a bucket **LS** (long-sentence) for length 40+, a single bucket **MS** (middle-sentence) for 16–40, and a single bucket **SS** (short-sentence) for 1–15. The length thresholds are tuning on the development data.
   - Number of unknown words: Words that never appear in the training set are unknown words. Sentences containing more unknown words are more unlikely to be parsed correctly. Since the number of unknown words usually ranges in a small interval, we didn't group them info buckets. Therefore, each number is assigned a single bucket.
     Note that the two features indicated above have the same value over the words in the same sentence.
   - Unknown word property: The boolean feature represents whether a word is an unknown word. The parser performance is worse for an unknown word.
   - Current word: $w_m$ (current word at position $m$).
   - Bigrams of POS tags: $t_{m-1}t_m$ (POS tags at position $m-1$ and $m$) and $t_m t_{m+1}$.
   - Trigram of POS tag: $t_{m-1}t_m t_{m+1}$.
2. Parser-based features are extracted based on parsing results.
   - The length of a dependency arc: Number of words between $w_m$ and $w_h$ (head word). As with the text-based "Length of Sentence" feature, we also attempted to group this feature into buckets. However, it did not help. So we just assign a single bucket for a single length.
   - Word collocation: $w_m w_h$ (current word at position $m$ and its head word).
   - POS tag collocation: $t_m t_h$ (POS of current word and its head word).
   - Dependency relation: $l_m$ (dependency relation label of current word and its head word). We have also considered the direction of the dependency arc with respect to the word, but it brings no improvement.
   - Combination of POS tag and dependency relation: $t_m l_m$.
   - The InBetween POS features (form of a POS trigram): $t_m t_b t_h$ (POS of current word, of its parent, and of a word in between)
   - The Surrounding POS features (form of a POS 4-gram): $t_{m-1}t_m t_h t_{h-1}$,$t_{m-1}t_m t_h t_{h+1}$, $t_{m+1}t_m t_h t_{h-1}$,$t_{m+1}t_m t_h t_{h+1}$.
   - Combined features. The combination of some of the features above can prove very helpful.
3. Comparison among Parsers
   - Agreement of different parsers: Intuitively, if two parsers based on different learning schemes agree on one dependency arc, it will probably be a reliable one. In this paper, we use a transition-based parser (Nivre, 2006) as the reference parser. If the reference parser products the same labeling (including the head and dependency label) for the current word with the basic parser, this feature is set TRUE; otherwise FALSE.

### 3.3. Training process

In this section, we will introduce how to train a RADAR model. Fig. 2 shows the training process flow. The core problem is how to construct a reliable arc training corpus. Here, $n$-fold validation method is used. At step ①, the traditional dependency parsing training corpus was divided into $n$ folds. At each time, $n$-1 folds were selected to train a dependency parser with graph-based mod-

[2] This is equivalent to find the correct head word and dependency relation for a word.

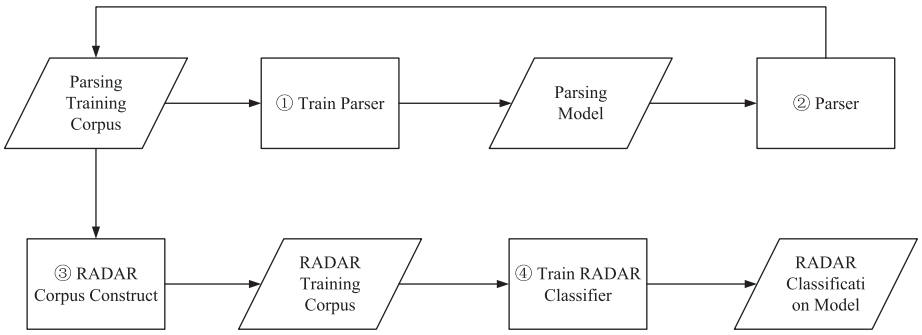


**Fig. 2.** The process flow of training a reliable arc classifier.

els. Then the leaving fold was parsed with the parser at step ② and the obtained automatic parsing dependency arcs were split into positive (correct) and negative (incorrect) classes according to their comparison with the gold parse trees. The above steps were repeated $n$ times and the RADAR training corpus was constructed at step ③. Finally at step ④, the reliable arc classifier was trained with the training corpus.

## 4. Experiments

### 4.1. Data set

In order to evaluate the effectiveness of the our approach, we conducted experiments on both English data and Chinese data.

For English, we used the data of CoNLL 2009 Shared Task: Syntactic Dependency Parsing (Hajič et al., 2009a). However, when dealing with sentences longer than 60 words, the 1000-best parsing process became extremely slow and memory consumption. Therefore, we just discard those sentences in our experiments. For Chinese, we used the Chinese Dependency Treebank (CDT) as the experimental data set (Liu, Ma, & Li, 2006). CDT consists of 60,000 sentences from the People's Daily in 1990s. The third and fourth columns of Table 1 show the numbers of sentences and dependency arcs (not including the punctuations) in training, development, and test set respectively.

### 4.2. Dependency parser and classifier

An open source NLP toolkit, mate-tools,[3] was selected as the dependency parser, which implemented a fast and the state-of-the-art graph-based dependency model (Bohnet, 2010). We modified the source code of mate-tools to output $k$-best results of each sentence. The probability of each dependency arc was calculated with the method introduced in Section 3.1. The last column of Table 1 shows the performance of mate-tools on the CoNLL and CDT data set. Note that the training data are constructed with a fourfold validation as described in Section 3.3 and the LAS of the training data is the average of these folds. Finally, the parser for the development and test data was trained on the whole training data. According to the LAS value as we can see, for English data, the number of positive examples rates about 8 to 9 times as the number of negative examples, while for Chinese data that is 4 to 5 times, which implies an imbalance.

The maltparser (Nivre et al., 2007) with default parameter settings was used as the transition-based reference parser to obtain the Agreement feature.[4]

The liblinear (Fan, Chang, Hsieh, Wang, & Lin, 2008) was used as the RADAR classifier,[5] which implements the L2-regularized logistic regression algorithm with parameters $C^+$ and $C^-$.

**Table 1**
Number of sentences and dependency arcs in training, development, and test set and their corresponding performance.

| Lang | Data set | #{sent} | #{arcs} | LAS (%) |
|---|---|---|---|---|
| Chinese | Train | 55,496 | 1,025,054 | 81.37 |
| | Dev | 1,500 | 29,091 | 81.19 |
| | Test | 3,000 | 56,786 | 81.43 |
| English | Train | 39,060 | 837,340 | 89.72 |
| | Dev | 1,325 | 29,024 | 88.36 |
| | Test | 2,389 | 50,291 | 90.03 |

**Table 2**
The contribution of different sort of features on the development data of CoNLL2009-en and CDT.

| Lang | Feature sort | AUC-PR (%) | Decrease |
|---|---|---|---|
| Chinese | All Features | 94.89 | N/A |
| | –Text-based | 94.81 | 0.07% |
| | –Parser-based | 93.06 | 1.75% |
| | –Comparison | 93.90 | 0.99% |
| English | All Features | 97.97 | N/A |
| | –Text-based | 97.95 | 0.02% |
| | –Parser-based | 97.35 | 0.62% |
| | –Comparison | 97.73 | 0.24% |

**Table 3**
AUC-PR and AUC-ROC measurements on the test data.

| Lang | Classification | | Probabilistic | |
|---|---|---|---|---|
| | AUC-PR (%) | AUC-ROC (%) | AUC-PR (%) | AUC-ROC (%) |
| Chinese | 94.65 | 81.42 | 93.59 | 79.38 |
| English | 98.56 | 89.29 | 97.68 | 85.50 |

### 4.3. Evaluation method

Accuracy is perhaps the most commonly used evaluation method for a classification problem. However, for imbalanced data, accuracy can yield misleading conclusions. For example, for a binary classification problem with a 90:10 class distribution, the Accuracy can easily reach 90% when we simply classify all the samples to the majority class. In practical applications, it is strongly desired that the most of used arcs are reliable (high precision). However, if

[3] http://code.google.com/p/mate-tools/.
[4] http://maltparser.org/.
[5] http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

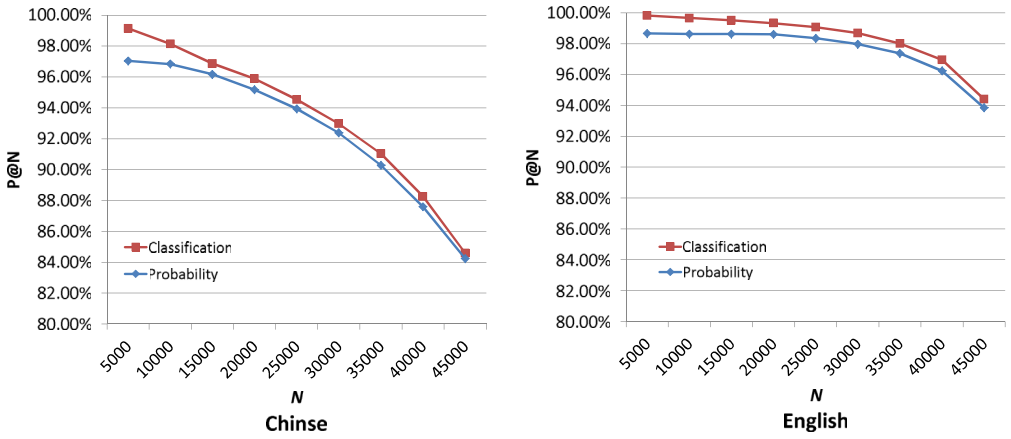


**Fig. 3.** P@N curves on test data set.

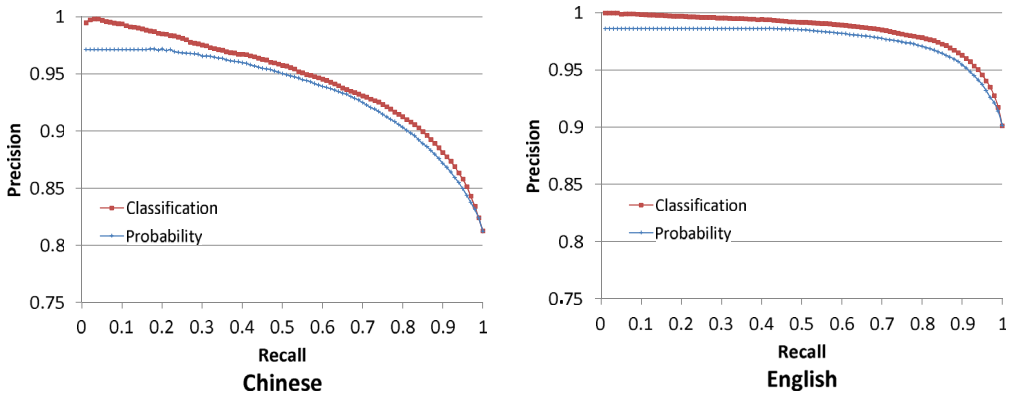


**Fig. 4.** PR curves on the test data.

studies only care about the *precision* of reliable arcs, the number of recognized reliable arcs can be restricted. This situation leads to a low recall of reliable arcs, which are useless in practical applications. The *F measure* is another normal evaluation method that considers precision and recall at the same time. However, it is not a proper evaluation here either. The same with accuracy as in the previous example, the minority class has much less impact on the *F score* than the majority class.

In practice, the top *N* most reliable dependency arcs are important. It can be evaluated with another information retrieval evaluation method, P@N. The *N* is different for different applications. For instance, for a search engine, the *N* is set to 10 or 20 usually, because users usually only check those top results. However, in the RADAR problem, the *N* should be larger, while it is difficult to fix the *N* for different applications.

In case of imbalanced data, ROC curve is usually thought as a suitable evaluation metric. However, it was suggested that the PR curves give more informative pictures of the performance of a learning algorithm (Davis & Goadrich, 2006). Therefore, in the current experiments, the PR curve is used as another evaluation method. In addition, we also calculate the Area Under both the ROC Curve and PR Curve in order to give a quantitative comparison. The AUCCalculator-0.2 tool was used to calculate the AUC-PR and AUC-ROC.[6] Larger AUC values indicate better performance.

[6] http://mark.goadrich.com/programs/AUC/.

### 4.4. Contribution of features

To evaluate the contribution of different sorts of features mentioned in Section 3.2.2, we build a classifier with all of these three sorts of features at first. Then, we threw away each sort of features respectively to see the performance decreasing. Once a sort of thrown features drop down the performance more than another, it indicates that this sort of features is more important than another. Contributions are evaluated on the development set of English data and Chinese data respectively. Table 2 shows the results. Note that in order to achieve optimal AUC-PR, the parameters of logistic regression classifier are tuned on the development data. The final settings are $C^{+} = 1$ and $C^{-} = 1.4$. Since $C^{-} > C^{+}$, we can prevent negative examples from being easily recognized into positive class.

From Table 2, we can see that all of the three sorts of features are useful for RADAR, because each removed sort of features decrease the performance. Among these three sorts, the parser-based features contribute the most and text-based features are the least important. The reason is probably that text-based features do not look at the full parse tree, they just evaluate the hardness of correctly parsing a sentence or a modifier. However, this is far from enough to decide the reliability of an arc. At the same time, some of the text-based features, such as the sentence length and the unknown word number, have the same value for all words in a sentence. As a consequence, they would be less indicative. In addition, The comparison features are also helpful.

### 4.5. Results

Table 3 shows the AUC-PR and AUC-ROC measurements on the test data. We can see that both for English and Chinese data, the classification method outperforms the probabilistic baseline system. In order to understand the improvement better and more intuitively, we provide the P@N curve with different $N$ in Fig. 3 and the Precision-Recall (PR) curve in Fig. 4.

We can see in the figures that, for different *Recall*/$N$, the classification method is all better than the probabilistic baseline, especially when $N$ is small or *Recall* is low. Based on our analysis, there are two reasons for this. First, the $k$-best approaching for estimating the probability of an arc is an approximate one. It depends very much on the selection of $k$. Although a 1000-best estimation is good enough, it is not ideal. The second factor is that the classification method can make use of more extra information (features) such as parsers' agreement, as well as more powerful tools (classifiers), which help to defeat the probabilistic method. We would also note that the $k$-best decoding in graph-based parsing comes at a high cost in terms of both memory and time consumption, especially when a large $k$ (e.g. $k = 500,1000\ldots$) is set. Thus the classification approach is much more efficient and flexible than the probabilistic approach.

## 5. Conclusion

This present paper proposes a novel natural language processing task, RADAR (ReliAble Dependency Arc Recognition). The performance of practical applications, such as information extraction and question answering, has potential to be improved further if the reliable arcs can be recognized correctly rather than only recognizing reliable parse trees of whole sentences.

We model RADAR as a binary classification problem with imbalanced data. Then we can elastically use various features and classifiers to achieve better performance. In this paper, we design three sorts of features to express reliability of arcs. A logistic regression model with adjustable $C$ parameter was used as the classifier, which can classify large scale data with high speed and performance.

Different from normal binary classification problem, RADAR cannot be evaluated with accuracy or *F measure* because of the problem of data imbalance. Instead, the P@N curve and PR curve, together with the use of associated area under the curve (AUC-PR, AUC-ROC) evaluate RADAR systems more suitably.

The experimental results show that the classification method with appropriate features can outperform the arc probabilistic baseline method. In addition, we also evaluated the contributions of different sorts features.

## 6. Future work

In the future, the work can be extended in the following ways:

1. The improvement of the performance of RADAR will be in two aspects, classifier and features. Besides logistic regression, more classifiers can be compared, which should not only have a high speed and accuracy, but also have output probability. RADAR also can be regarded as a ranking problem and can use various learning to rank algorithms. Besides classification methods, more combinatorial features and global features can be combined, such as the reliabilities surrounding the other and current arcs. Hence, it is no longer a simple binary classification model. In addition, another kind of global features counts from large scale unlabeled data, such as the number of arcs, which are parsed automatically or the Google Web 1T n-gram count information can be used.
2. Apply RADAR to applications, such as entity relation extraction based on a web corpus. These applications may recognize larger reliable sub-structures, such as parse paths. However, corresponding reliable arcs cannot be accumulated simply. Thus, recognizing reliable sub-structures becomes an interesting problem.
3. It would be necessary to apply our approach to transition-based parser, to check whether it produces similar results.
4. The last future work is domain adaption problem, i.e. how to adapt the reliability model trained on one domain into another domain or even open domains. This is also an open question for natural language processing.

## Acknowledgment

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via Grant 61133012 and 61370164. The National Basic Research Program (973 Program) of China via Grant 2014CB340503.

## References

Androutsopoulos, I., & Malakasiotis, P. (2010). A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research, 38*(1), 135–187.

Atserias, J., Attardi, G., Simi, M., & Zaragoza, H. (2010). *Active learning for building a corpus of questions for parsing. LREC*. European Language Resources Association.

Avihai Mejer, K. C. (2012). Are you sure? Confidence in prediction of dependency tree edges. In *NAACL-HLT2012*.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007). Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence. IJCAI'07* (pp. 2670–2676). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Bishop, C. M. (2006). *Pattern recognition and machine learning (Information science and statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc..

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, August 2010. (pp. 89–97).

Chen, W., Kawahara, D., Uchimoto, K., & Zhang, Y. (2008). Dependency parsing with short dependency relations in unlabeled data. In *IJCNLP08*.

Chen, W., Kazama, J., Uchimoto, K., & Torisawa, K. (2009). Improving dependency parsing with subtrees from auto-parsed data. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing – EMNLP '09* (vol. 2). Morristown, NJ, USA: Association for Computational Linguistics, p. 570.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning. ICML '06* (pp. 233–240). New York, NY, USA: ACM.

Dell'Orletta, F., & Venturi, G. (2011). ULISSE: An unsupervised algorithm for detecting reliable dependency parses. In *CoNLL11* (pp. 115–124).

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research, 9*, 1871–1874.

Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics, 28*, 245–288.

Goldwasser, D., Reichart, R., Clarke, J., & Roth, D. (2011). Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 1486–1495). Portland, Oregon, USA: Association for Computational Linguistics.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., et al. (2009a). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task* (pp. 1–18). Boulder, Colorado: Association for Computational Linguistics.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M.A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., & Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL-2009*.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia.

Kawahara, D., & Kurohashi, S. (2010). Acquiring reliable predicate-argument structures from raw corpora for case frame compilation. In *LREC10* (pp. 1389–1393).

Kawahara, D., & Uchimoto, K. (2008). Learning reliability of parses for domain adaptation of dependency parsing. In *IJCNLP08* (pp. 709–714).

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., & Tsujii, J. (2009). Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task. BioNLP '09* (pp. 1–9). Stroudsburg, PA, USA: Association for Computational Linguistics.

Koo, T., Globerson, A., Carreras, X., & Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 141–150). Prague, Czech Republic: Association for Computational Linguistics.

Kübler, S., McDonald, R. T., & Nivre, J. (2009). *Dependency parsing. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Liu, T., Ma, J., & Li, S. (2006). Building a dependency treebank for improving chinese parser. *Journal of Chinese Language and Computing, 16*, 207–224.

Meena, A., & Prabhakar, T. V. (2007). Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis. In *Proceedings of the 29th European Conference on IR Research. ECIR'07* (pp. 573–580). Berlin, Heidelberg: Springer-Verlag.

Nivre, J. (2006). *Inductive dependency parsing (Text, Speech and Language Technology)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc..

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., et al. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering, 13*(2), 95–135.

Ravi, S., Knight, K., & Soricut, R. (2008). Automatic prediction of parser accuracy. In *EMNLP08* (pp. 887–896). Morristown, NJ, USA: Association for Computational Linguistics.

Reichart, R., & Rappoport, A. (2007). An ensemble method for selection of high quality parses. In *ACL07* (pp. 408–415).

Sag, I. A., Baldwin, T., Bond, F., Copestake, A. A., & Flickinger, D. (2002). Multiword expressions: A pain in the neck for nlp. In *Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Text Processing. CICLing '02* (pp. 1–15). London, UK, UK: Springer-Verlag.

Settles, B. (2010). Active learning literature survey. Tech. rep., University of Wisconsin-Madison.

van Noord, G. (2007). Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies – IWPT '07* (pp. 1–10). Morristown, NJ, USA: Association for Computational Linguistics.

Wu, F., & Weld, D. S. (2010). Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. ACL '10* (pp. 118–127). Stroudsburg, PA, USA: Association for Computational Linguistics.

Yates, A., Schoenmackers, S., & Etzioni, O. (2006). Detecting parser errors using web-based semantic filters. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing – EMNLP '06* (pp. 27). Morristown, NJ, USA: Association for Computational Linguistics.

Zhang, M., Zhang, J., Su, J., & Zhou, G. (2006). A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. ACL-44* (pp. 825–832). Stroudsburg, PA, USA: Association for Computational Linguistics.