# Improving Unsupervised Dependency Parsing with Knowledge from Query Logs

XIUMING QIAO, HAILONG CAO, and TIEJUN ZHAO, Harbin Institute of Technology

3

Unsupervised dependency parsing becomes more and more popular in recent years because it does not need expensive annotations, such as treebanks, which are required for supervised and semi-supervised dependency parsing. However, its accuracy is still far below that of supervised dependency parsers, partly due to the fact that their parsing model is insufficient to capture linguistic phenomena underlying texts. The performance for unsupervised dependency parsing can be improved by mining knowledge from the texts and by incorporating it into the model. In this article, syntactic knowledge is acquired from query logs to help estimate better probabilities in dependency models with valence. The proposed method is language independent and obtains an improvement of 4.1% unlabeled accuracy on the Penn Chinese Treebank by utilizing additional dependency relations from the Sogou query logs and Baidu query logs. Morever, experiments show that the proposed model achieves improvements of 8.07% on CoNLL 2007 English using the AOL query logs. We believe query logs are useful sources of syntactic knowledge for many natural language processing (NLP) tasks.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Unsupervised learning**

Additional Key Words and Phrases: Dependency parsing, query logs, additional knowledge, natural annotations

**ACM Reference Format:**
Xiuming Qiao, Hailong Cao, and Tiejun Zhao. 2016. Improving unsupervised dependency parsing with knowledge from query logs. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 16, 1, Article 3 (June 2016), 12 pages.
DOI: http://dx.doi.org/10.1145/2903720

## 1. INTRODUCTION

Dependency parsing is the task of analyzing dependency relations (*head* → *dependent*) between words in one sentence. It is widely applied in machine translation [Quirk et al. 2005; Shen et al. 2010], information extraction [Culotta and Sorensen 2004], question answering [Cui et al. 2005; Wang et al. 2007], and so on. Dependency parser can be either trained in a supervised, semi-supervised, or unsupervised fashion. Supervised dependency parsing [McDonald and Pereira 2006; Nivre et al. 2007] and semi-supervised dependency parsing [Koo et al. 2008; Chen et al. 2013] can achieve high performance, but they rely too much on treebank annotations. Treebanks are difficult and expensive to build. What's more, most of the existing treebanks are concentrated on the news

The work of this article was funded by the project of National Natural Science Foundation of China (Grants No. 91520204 and No. 61572154) and the project of National High Technology Research and Development Program of China (863 Program) (Grant No. 2015AA015405).
Authors' addresses: X. Qiao, H. Cao (corresponding author), and T. Zhao, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China; emails: {xmqiao, hailong, tjzhao}@mtlab.hit.edu.cn.

DOI: http://dx.doi.org/10.1145/2903720

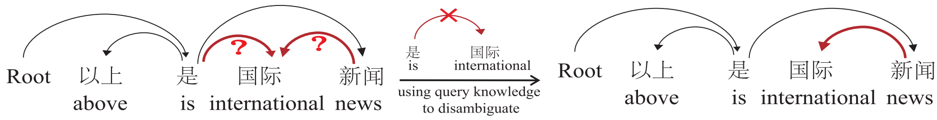


Fig. 1. An example of using knowledge from queries to disambiguate.

domain [Yu et al. 2013]. Therefore, more and more people tend to research unsupervised dependency parsing.

Unsupervised dependency parsing is appealing since it does not need annotated treebank and can adapt to any domain. However, its accuracy is very low due to lack of sufficient knowledge. Recent researches show that additional knowledge such as punctuation [Spitkovsky et al. 2011], word cluster [Spitkovsky et al. 2011], lexical [Headden 2012], reducibility [Mareček and Straka 2013], and bilingual information [Liu et al. 2013; Ma and Xia 2014] are very effective to improve unsupervised dependency parsing. Alternatively, in this article, we propose a simple yet very effective approach to improve unsupervised dependency parsing by making use of query logs, which are available for many languages, even on a large scale. Queries are input by users when interacting with search engines. Although each query is short and contains only few words, the words in a query are not independent with each other. For example, if someone wants to browse international news, he/she usually enters "国际/international 新闻/news" on the search engine, which is a noun-modifier structure. Actually, according to our manual evaluation on the query logs (see Section 3.1), we find that about 76% of the queries contain syntactic structures. These syntactic structures could be naturally used as human annotated data to disambiguate for unsupervised dependency parsing. By using the dependency structure "国际/international 新闻/news" as a (soft) constraint, one can easily imagine that "国际/international" should not be attached into "是/is" but be attached into "新闻/news" when parsing the sentence "以上/above 是/is 国际/international 新闻/news" (denoted as $s_1$), as shown in Figure 1.

In this article, we employ two steps to put the above idea into practice. First, we automatically acquire dependency structures from query logs and define a syntactic relation score model over a pair of dependent words based on their occurrence. Given such a model, even though the exact kind of dependency structures in a query is unclear, we can still know how probable one word depends on another. We make three main contributions in this article:

—We publish manually annotated query log data[1] that may be useful for other natural language processing (NLP) tasks such as semantic parsing (Section 3.1).
—We show an approach to acquire syntactic knowledge from query logs for dependency parsing (Section 3.2).
—We propose Query-Augmented Dependency Model with Valence (QA-DMV) (Section 4), which obtains substantial improvements over the standard dependency model with valence on both the Chinese Penn Treebank and the CoNLL 2007 English task (Section 5).

## 2. DEPENDENCY MODEL WITH VALENCE

One of the most successful unsupervised dependency parsing models is the DMV, which uses only part-of-speech (POS) tags [Klein and Manning 2004]. It is a generative model in which a dependency tree $T$ is generated from a given sentence $S$ by maximizing the

[1]These data are available in https://github.com/hitxiaoqiao/data-for-linguistic-analysis-of-queries.git. We use these data to make a linguistic analysis of query logs.

conditional probability $P(T|S)$:

$$P(T|S) = \frac{P(T,S)}{P(S)} \propto P(T,S). \tag{1}$$

Since $P(S)$ is a constant, our maximization problem can be regarded as a joint inference of $P(T,S)$. Moreover, our problem can be reduced to the maximization problem of $P(T)$, given $P(T,S) = P(T)P(S|T)$, and $S$ is the leaf string of $T$.

First, the root of $T$ is selected according to the probability of selecting a position in the rightward direction. Then, its children are generated to the left of the root until we make a decision to stop. Similarly, the children to the right of root are generated until we decide to stop. The process is recursively performed for each generated word until we have generated all the leaves. Whether to generate a child ($\neg stop$) or not ($stop$) for $h$ (head word) in the direction of $dir$ is decided by $P_{stop}(\neg stop|h, dir, adj, f)$ and $P_{stop}(stop|h, dir, adj, f)$, where $dir$ is left or right, $adj$ is a binary variable indicating whether $h$ already has a child in the direction of $dir$, and $f$ is the list of children of $h$ in the direction of $dir$. If stop, then no more children of $h$ in the direction $dir$ will be generated. If not, then a child $d$ is chosen according to the attach-probability $P_{attach}(d|h, dir)$[2]. For the child $d$, its subtree is generated recursively in a similar way.

We denote a subtree of $T$ as $D$. $D$ has left children $deps(h,l)$ and right children $deps(h,r)$. Then, the probability for the subtree tree $D(h)$, $P_{tree}(D(h))$, is recursively computed as follows:

$$P_{tree}(D(h)) = \prod_{dir \in l,r} \prod_{d \in deps(h,dir)} P_{stop}(\neg stop|h, dir, adj, f) \cdot P_{attach}(d|h, dir) \cdot P_{tree}(d)$$
$$P_{stop}(stop|h, dir, adj, f). \tag{2}$$

The probability of a single tree $P_{tree}(T)$ is computed as follows:

$$P_{tree}(T) = P_{attach}(head(T)|ROOT, right) \cdot P_{tree}(D(head(T))). \tag{3}$$

## 3. SYNTACTIC KNOWLEDGE FROM QUERY LOGS

We assume that queries are not merely short plain texts but contain latent syntactic structures. Our goal is to mine such syntactic knowledge implicated in each query. We use a score function $score(x, y)$ to measure the relation between two words $x$ and $y$ occurring in queries.

### 3.1. Linguistic Analysis of Query Logs

A query is manually input and can reflect a human's consciousness using a few words. Indeed, our preliminary studies indicate that syntactic structures are preserved in many queries, such as predicate-object, subject-predicate, or noun-modifier. For example, "Ctex download" is a predicate-object structure, "Jackson dance" is a subject-predicate structure, and "birthday cakes" is a noun-modifier structure. These structures may be substructures of many sentences.

We annotate the syntactic structures of 300 queries, and Table I shows the ratio of each relation, where "Others" denotes those queries do not have syntactic relations. For example, there seem to be no syntactic relations among the words in the query of "Bieber twitter." Table I indicates that queries that contain syntactic structures account for about 76% of all 300 queries. Query log is rich with syntactic knowledge.

Though we cannot distinguish which dependency relation within queries in unsupervised setting, we can get the strength of the syntactic relation between two words

[2]The attach probability is also denoted as choose probability.

Table I. Ratio of Each Kind of Structures in 300 Queries

| Relation | Explanation | Ratio |
|---|---|---|
| NMOD | noun modifier | 62.21% |
| SUB | subject-predicate | 4.68% |
| OBJ | predicate-object | 4.01% |
| VMOD | verb modifier | 4.01% |
| VC | verb chain | 1.34% |
| Others | other structures | 24.08% |

through computing their occurrence, and the "strength" means the size of the probability that one word depends on another word.

### 3.2. Syntactic Relation Score

Let $score(x, y)$ denote the strength of the syntactic relation between two words $x$ and $y$ acquired from query logs. $count(x)$ denotes the number of times word $x$ occurs in query logs, and $count(xy)$ denotes the number[3] of times $x$ and $y$ co-occur in query logs. The score function should satisfy these three constraints:

(1) Its value is between 0 ($x$ and $y$ never co-occur) and 1 ($x$ and $y$ always co-occur);
(2) It is symmetrical for each word;
(3) If $count(xy) > count(zy)$ and $count(x) = count(z)$, then $score(x, y) > score(z, y)$.

Our definition of $score(x, y)$ is similar to pointwise mutual information [Church and Hanks 1990], except that its value is between 0 and 1. The second constraint is set because we find that words in queries are always unordered, especially for the queries that contain only two words. In this extreme case, we can easily predict that two words are more likely a single head-dependent pair, but still we cannot tell which one is a head. The third constraint implicates that $score(x, y)$ is decided by $count(xy)$, $count(x)$, and $count(y)$ together.

Under the above constraints, the score of $x$ and $y$ can be computed as follows:

$$score(x, y) = \frac{1}{2} \cdot \left( \frac{count(xy)}{count(x)} + \frac{count(xy)}{count(y)} \right). \tag{4}$$

## 4. MODEL

Now we will introduce our model, QA-DMV, a query-augmented DMV, and an extension of DMV using the syntactic relationships acquired from query logs.

### 4.1. Query-Augmented Dependency Model with Valence

In the original DMV framework, the model incorporates only POS tags and completely ignores lexical features. Our QA-DMV indirectly augments DMV with lexical features using the head-dependent relationship score estimated from query logs through linear interpolation that is similar to that in Headden [2012].

The model of $P^{dmv}_{stop}(\neg stop|h, dir, adj, f)$ decides whether to generate another child of $h$,

$$P^{dmv}_{stop}(\neg stop|h, dir, adj, f) = 1 - P^{dmv}_{stop}(stop|h, dir, adj, f). \tag{5}$$

In QA-DMV, the stop-probability is computed based on the POS tags, not on the surface word, as follows [Mareček and Straka 2013]:

$$P^{dmv}_{stop}(stop|h, dir, adj, f) = \frac{\frac{2}{3} + count(stop, c_h, dir, adj, c_f)}{1 + count(c_h, dir, adj, c_f)}, \tag{6}$$

[3]We ignore the ordering between $x$ and $y$, because there is no order feature in our baseline model.

where $c_h$ is the POS tag of $h$, and $c_f$ is the corresponding POS tag list of $f$. $count(stop, c_h, dir, adj, c_f)$ is the frequency of selecting $c_f$ as the dependents of $c_h$, and no more other dependents in the direction $dir$. Similarly, $count(c_h, dir, adj, c_f)$ is the frequency of selecting $c_f$ as dependents for $c_h$ in the direction $dir$. We smooth it with the parameter 2/3 following Mareček and Straka [2013]. If $h$ has no dependent in the current direction $dir$, then $adj$ is 1. Otherwise, if $h$ has one or more dependents, then $adj$ is 0.

The attachment model $P^{dmv}_{attach}(d|h, dir)$ decides the dependent $d$ for $h$ in the direction dir, and it is computed as follows in the original framework [Mareček and Straka 2013]:

$$P^{dmv}_{attach}(d|h, dir) = \frac{\frac{\alpha_c}{|C|} + count(c_d, c_h, dir)}{|C| + count(c_h, dir)}, \tag{7}$$

where $count(c_d, c_h, dir)$ denotes the frequency of choosing $c_d$ as a dependent of $c_h$ in direction $dir$, and $count(c_h, dir)$ is the frequency of $c_h$ being a head in direction $dir$. Following the suggestions by Mareček and Straka [2013], the attachment-probability is smoothed by $\alpha_c$ and $|C|$ where $|C|$ is the number of POS tag categories in the whole corpus, and $\alpha_c$ is empirically set to 50.

In order to get the relative attach probability model estimated from query logs, we normalize the syntactic relation score in Equation (4) as follows:

$$P^{query}_{attach}(d|h) = \frac{score(d, h)}{\sum_{d_i \in C} score(d_i, h)}. \tag{8}$$

The model is linearly interpolated with the attachment model in Equation (7) and we obtain $P^{dmv+query}_{attach}(d|h, dir)$ as follows:

$$\begin{aligned} & if \quad h \in Q \quad and \quad d \in Q, then: \\ & \quad P^{dmv+query}_{attach}(d|h, dir) = \gamma \cdot P^{query}_{attach}(d|h) + (1-\gamma) \cdot P^{dmv}_{attach}(d|h, dir) \\ & else: \\ & \quad P^{dmv+query}_{attach}(d|h, dir) = P^{dmv}_{attach}(d|h, dir), \end{aligned} \tag{9}$$

where $Q$ is the set of words found in query logs. Note that the syntactic relation model of Equation (8) is selectively interpolated with the original attachment model of Equation (7) using a constant $\gamma$ $(0 \le \gamma \le 1)$ in order to avoid over penalties when the words h and d are never observed in $Q$. When $\gamma = 0$, we uncover the baseline model that does not consider the syntactic relations estimated from query logs. Similarly, when $\gamma = 1$, only the syntactic relations are employed to assign the attachment probabilities when two words are found in $Q$. The probability for the subtree $D(h)$ is computed as follows:

$$\begin{aligned} P_{tree}(D(h)) = \prod_{dir \in l,r} \prod_{d \in deps(h,dir)} & P^{dmv}_{stop}(\neg stop|h, dir, adj, f) \cdot P^{dmv+query}_{attach}(d|h, dir) \cdot P_{tree}(d) \\ & P^{dmv}_{stop}(stop|h, dir, adj, f). \end{aligned} \tag{10}$$

Finally, $P_{tree}(T)$, the probability of the whole tree $T$ is computed in the same way as Equation (3). The probability of a treebank $P_{treebank}$ is computed as the product of the probabilities of all trees in the treebank:

$$P_{treebank} = \prod_{T \in treebank} P_{tree}(T). \tag{11}$$

Table II. Data Sets (in Sentence Number)

| Data | Train | Dev | Test |
|---|---|---|---|
| CTB5 | 16091 | 803 | 1910 |
| CTB5($\leq$10) | 3951 | 205 | 486 |
| CoNLL07 English | 18577 | — | 214 |

### 4.2. Inference

Our parser is a variation of the parser in Mareček and Straka [2013], which differs in the computation methods of stop probability and attach probability. The inference steps of our parser are as follows:

—Initialization: A random projective dependency structure is given to each sentence.
—Sampling: We use Gibbs sampling to sample a dependency tree for each sentence, according to other annotated sentences. We believe that the corpus is large enough to ignore the impact of edges within the same sentence.
—Sampling is done iteratively until the convergence of $P_{treebank}$. After the burn-in period (first 500 iterations), we count every edge $e$ in sentences from the 501th iteration to the 1000th iteration and save it in the format of $count(e)$, following Mareček and Straka [2013].
—Decoding: Chu-Liu/Edmond's algorithm [Chu and Liu 1965] is used to decode every sentence.

The count of each edge gained in sampling is used as its weight, that is, $count(e)$. The maximum spanning tree we need is the tree that maximize the sum of weights for all $e \in T$.

$$T_{mst} = \arg\max_{T} \sum_{e \in T} count(e). \tag{12}$$

## 5. EXPERIMENTS AND RESULTS

### 5.1. Data

We use SogouQ query logs (Version 2008)[4] and Baidu query logs (part of queries in a month of 2010)[5] as our Chinese knowledge source. The SogouQ contains 44 million (M) queries of March 2007 [Liu et al. 2011], and the Baidu query logs contain 108M queries.

The above Chinese queries are simplified Chinese texts, so we evaluate our parser on the Penn Chinese Treebank 5 (CTB5). We adopt the data split of Li et al. [2014] and we use Penn2Malt[6] to convert the original constituency trees into dependency trees with its default head rules. Table II shows the data statistics. The coverage of words in Sogou and Baidu queries over CTB5 is 57.48%.

We also conduct experiments on English. AOL query logs[7] are used as English knowledge source. This query set contains 36M queries.

CoNLL07 English is the dependency data used in CoNLL 2007 shared task on dependency parsing. Its training set is WSJ sections 2-11 of Penn Treebank and the test set is a subset of section 23. The data used in many state-of-the-art related works are CoNLL07 English. To better compare with other works, we use CoNLL07 English to evaluate our parser. The coverage of words in AOL query logs over CoNLL 2007 English is 43.79%.

[4]http://www.sogou.com/labs/dl/q.html.
[5]http://openresearch.baidu.com/.
[6]http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html.
[7]http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/.

Table III. Tuning $\gamma$ on CTB5 Development Set

| $\gamma$ | baseline | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **UAS** | 40.93 | 41.22 | 41.03 | 41.69 | 42.45 | 42.73 | 41.97 | 43.40 | 44.44 | 44.82 | 43.11 |

We process the query logs as follows: Extract user queries and remove other information such as ID, URL and so on; remove queries containing non-Chinese (or non-English) characters; segment queries by Stanford segmenter[8]; and keep queries that only contain two words. If we have more query logs in other languages, then we will experiment in other datasets of CoNLL 2007 in future work.

### 5.2. Baseline Parser

Our baseline parser is a unsupervised dependency parser[9] with a pure DMV [Mareček and Straka 2013]. The parser uses the original stop probabilities and the attachment probabilities as described in Equation (6) and Equation (7). This parser uses Gibbs sampling as the training method and samples in many iterations until the convergence of $P_{treebank}$. But the Gibbs sampler does not always convergent on a similar grammar, so we run each inference 50 times and take the run with the highest $P_{treebank}$ for the evaluation, following Mareček and Straka [2013]. We evaluate the parser by unlabeled attachment score (UAS): the percentage of words that have correct heads, excluding punctuations.

### 5.3. Parameter

During tuning $\gamma$ in Equation (9), we use SogouQ and Baidu query logs as Chinese knowledge source and test on the dev set of CTB5. Word pairs from the preprocessed queries are used to compute the model of $P_{attach}^{query}(d|h)$ in Equation (8). When $\gamma$ is 0.9, our parser performs best on development data, as shown in Table III. Then we adopt this setting in following evaluations.

### 5.4. Results

We evaluate our parser on the test set of CTB5, and the UAS is 46.31%, achieving improvement of 4.1% from the baseline system (DMV). We use syntactic knowledge from AOL queries to improve dependency parsing on English. When we evaluate our parser on the test set of CoNLL07 English, the UAS is 44.38%, winning the baseline system by 8.07%.

Table IV shows the comparison between performance of our parser and previous work on the Chinese Penn Treebank. Our parser is the QA-DMV with the best setting on development set. Klein and Manning [2004] implement an original DMV parser. Liu et al. [2013] use a bilingually guided parsing model. From Table IV, we can see that our parser performs better than other three parsers.

Table V shows the comparison between performance of our parser and previous work on CoNLL07 English test data. Mareček and Straka [2013] estimate stop probabilities from Wikipedia articles. Mareček and Žabokrtský [2012a] computes reducibility scores from Wikipedia articles. Spitkovsky et al. [2012] use different boundaries to help unsupervised dependency parsing.

Our parser does not perform better than Mareček and Žabokrtský [2012a] and Mareček and Straka [2013], which use POS tag reducibility gained from W2C corpus of Wikipedia articles [Majliš and Žabokrtský 2012]. However, the word coverage of

[8]http://nlp.stanford.edu/software/segmenter.shtml.
[9]http://ufal.mff.cuni.cz/udp.

Table IV. UAS Comparison on Chinese Penn Treebank

| System | UAS |
|---|---|
| Baseline Parser [length ≤ 10] | 42.21 |
| **Our Parser [length ≤ 10]** | 46.31 |
| Klein and Manning [2004] [length ≤ 10] | 42.5 |
| Liu et al. [2013] [all] | 22.6 |
| **Our Parser [all]** | 24.38 |

Table V. UAS Comparison on Chinese Penn Treebank

| System | UAS |
|---|---|
| Baseline Parser | 36.31 |
| **Our Parser** | 44.38 |
| Spitkovsky et al. [2012] | 29.2 |
| Mareček and Žabokrtský [2012a] | 49.2 |
| Mareček and Straka [2013] | 55.4 |

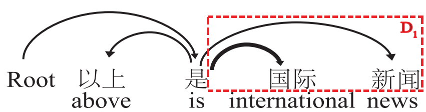


Fig. 2. The result of baseline for $s_1$.

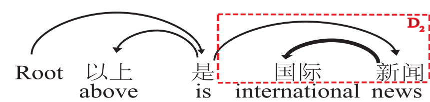


Fig. 3. Our result for $s_1$.

Table VI. The Running Time of Baseline and Our Parser (Per Word)

| System | Baseline | Our Method |
|---|---|---|
| **Running time(s)** | 0.009428 | 0.012124 |

AOL queries on CoNLL07 English is 43.79% and our performance will increase with the wider lexical coverage ratio, as discussed in Section 5.7.

We just use lexical surface features gained from queries rather than POS tags or cluster features, because Barr et al. [2008] points that about 70% of queries are noun phrases. And our experiments, which are not shown due to limited space, prove that the noise of POS tags and cluster information in queries are very big.

### 5.5. Analysis

As an error analysis, we show the parsing result by the baseline parser and our proposed parser in Figures 2 and 3, respectively, for the sentence in Figure 1. Although both results share the same left subtree of "是/is," they differ considerably in the right subtree, denoted by $D_1$ for the baseline and by $D_2$ for our proposed method.

Our syntactic relation model from the SogouQ and Baidu query logs indicates that

$$P^{query}_{attach}(\text{国际/international} \,|\, \text{是/is}) = 0, \qquad P^{query}_{attach}(\text{国际/international} \,|\, \text{新闻/news}) = 0.12$$

that is, the word "国际/international" has a stronger relation with the word "新闻/news" than with the word "是/is." During sampling, the right subtree of "是/is" is more likely to be sampled as $D_2$ rather than $D_1$. As a result, our parser tends to get the gold tree.

### 5.6. Running Time

In order to test whether the model of $P^{query}_{attach}(d|h)$ will affect parser's speed, we compare the running time of the baseline parser and our system with the same settings and corpus. We use AOL queries as a knowledge source and CoNLL 2007 English (≤10) as a test corpus. The number of iterations is set at 1000. The data of CoNLL 2007 English (≤10) has 18,916 words. The average run time per word of baseline parser and our parser is shown in Table VI. Our proposed method incurs an additional 0.0027s per word to the baseline parser, which is not extremely high.

### 5.7. The Impact of Query Logs' Scale

Moreover, we measure the impact of query size on parsing performance. Figure 4 shows the lexical coverage ratio of different size of queries on CoNLL 2007 English (≤10) data.

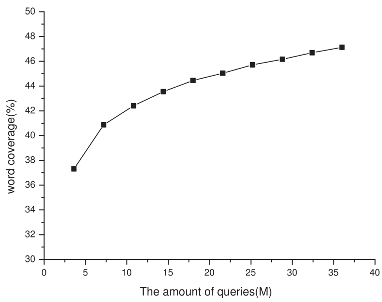


Fig. 4. The word coverage ratio of different scale queries on CoNLL 2007 English($\leq$10).

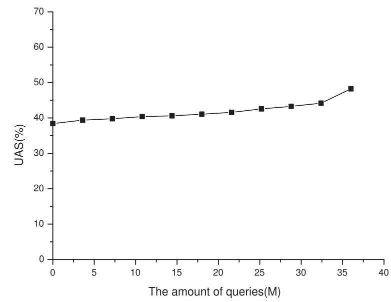


Fig. 5. The effect of the query scale on UAS.

Each dataset is a subset of larger sets. Figure 5 presents the experimental results on varying the AOL query log data size tested on CoNLL 2007 English ($\leq$10) data. The plot clearly shows that more query logs are helpful for better unsupervised parsing performance.

## 6. RELATED WORK

Previous work on unsupervised dependency parsing extends DMV by adding additional knowledge. Spitkovsky et al. [2010b] acquires natural annotations from web structure data (anchors, bold, italics, and underlines) and applies this information into decoding. The reducibility feature of one word [Mareček and Žabokrtský 2012b] or a word sequence [Mareček and Žabokrtský 2012a; Mareček and Straka 2013] is used in DMV and leads to many improvements on many languages. We propose to augment DMV with the syntactic knowledge from query logs. We assume that the words in each query are not independent with each other but contain latent structures that could be useful as a clue to find head-dependent relationship among texts.

Multilingual information [Cohen et al. 2011; Søgaard 2011] also works in unsupervised dependency parsing. Naseem et al. [2012] uses annotations from a diverse set of source languages, performing well in multi-source transfer-based dependency parsing. Liu et al. [2013] utilize information from both sides of bilingual corpus, outperforming previous bilingual-guided unsupervised models. Ma and Xia [2014] train parsing models for resource-poor languages by transferring cross-lingual knowledge from resource-rich languages with entropy regularization. The resources they use are limited, but the amount of query logs we can use is huge, because users will input more than 1 million queries everyday through the Sogou search engine. And there are many other search engines. Then we can cover more words and have a more accurate parser.

Much more information can be used to improve unsupervised dependency parsing, such as cluster information [Spitkovsky et al. 2011], lexicals [Headden 2012], punctuations [Spitkovsky et al. 2011], and sentence boundaries [Spitkovsky et al. 2012].

Moreover, many researchers are devoted to improving the training method in DMV. Klein and Manning [2004] use an inside-outside re-estimation method to learn the grammar without any smoothing. Headden et al. [2009] adds smoothing into DMV with rich context information. Spitkovsky et al. [2010] combines "Baby Steps" and "Less is More." Spitkovsky et al. [2010a] uses Viterbi EM to learn grammar, performing better in long sentences than classic EM.

Though simple, query logs contain many kinds of knowledge. Tannebaum and Rauber [2012] acquire lexical knowledge from query logs to help query expansion in patent searching. Li [2010] improves query understanding using its lexical features, syntactic features, and semantic features. Sekine and Suzuki [2007] extract ontological knowledge using search query logs. Tur et al. [2011] exploit user queries mined from search engine query click logs to bootstrap or improve slot filling models for spoken language understanding. To the best of our knowledge, our method is the first to incorporate syntactic knowledge from query logs into unsupervised dependency parsing.

## 7. CONCLUSION AND FUTURE WORK

In this article, we extracted syntactic knowledge from query logs to help estimate the attach probability in DMV. We presented significant improvements on Chinese and English unsupervised parsing task and also demonstrated that more queries can lead to better performance.

In the future, we will apply knowledge from query logs in other formats to further improve dependency parsing. Moreover, dependency parsing on queries is very important for information retrieval and its accuracy is rather low now. We will pay much more attention to query-dependency parsing.

## ACKNOWLEDGMENTS

We thank Dr. Taro Watanabe, Dr. Lemao Liu, and Xiaoning Zhu for their valuable discussions and thank reviewers for many valuable comments and helpful suggestions.

## REFERENCES

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *Proceedings of EMNLP 2008*. Association for Computational Linguistics, 1021–1030.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-supervised feature transformation for dependency parsing. In *EMNLP 2013*. Association for Computational Linguistics, Seattle, WA, 1303–1313.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Sci. Sinica* 14 (1965), 1396–1400.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.* 16, 1 (March 1990), 22–29.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the EMNLP 2011*. Association for Computational Linguistics, Edinburgh, Scotland, UK, 50–61.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR 2005*. ACM, New York, NY, 400–407.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL'04*. Barcelona, Spain, 423–429.

William P. Headden, III. 2012. *Unsupervised Bayesian Lexicalized Dependency Grammar Induction*. Ph.D. Dissertation. Brown University.

William P. Headden, III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL 2009*. Association for Computational Linguistics, Boulder, CO, 101–109.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL'04*. Association for Computational Linguistics, Article 478.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. ACL/HLT*.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of ACL '10*. Association for Computational Linguistics, 1337–1345.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the ACL*. Association for Computational Linguistics, Baltimore, MD, 457–467.

Kai Liu, Yajuan Lü, Wenbin Jiang, and Qun Liu. 2013. Bilingually-guided monolingual dependency grammar induction. In *Proceedings of ACL 2013*. Association for Computational Linguistics, Sofia, Bulgaria, 1063–1072.

Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, and Liyun Ru. 2011. How do users describe their information need: Query recommendation based on snippet click model. *Expert Syst. Appl.* 38, 11 (2011), 13847–13856. DOI:http://dx.doi.org/10.1016/j.eswa.2011.04.188

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL 2014*. Association for Computational Linguistics, Baltimore, MD, 1337–1348.

Martin Majliš and Zdeněk Žabokrtský. 2012. Language richness of the web. In *Proceedings of LREC-2012*. European Language Resources Association (ELRA), Istanbul, Turkey, 2927–2934. ACL Anthology Identifier: L12-1110.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of ACL'13*. Association for Computational Linguistics, Sofia, Bulgaria, 281–290.

David Mareček and Zdeněk Žabokrtský. 2012a. Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of EMNLP-CoNLL'12*. Association for Computational Linguistics, Jeju Island, Korea, 297–307.

David Mareček and Zdeněk Žabokrtský. 2012b. Unsupervised dependency parsing using reducibility and fertility features. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure (WILS'12)*. Association for Computational Linguistics, Stroudsburg, PA, 84–89.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics: EACL 2006 (EACL'06)*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of ACL'12*. Association for Computational Linguistics, Jeju Island, Korea, 629–637.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natur. Lang. Eng.* 13, 2 (2007), 95–135.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL 2005*. Association for Computational Linguistics, Ann Arbor, MI, 271–279.

Satoshi Sekine and Hisami Suzuki. 2007. Acquiring ontological knowledge from query logs. In *Proceedings of WWW'07*. ACM, New York, NY, 1223–1224.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Comput. Linguist.* 36, 4 (Dec. 2010), 649–671. DOI:http://dx.doi.org/10.1162/coli_a_00015

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, OR, 682–686.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of EMNLP 2011*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL-2011)*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Three dependency-and-boundary models for grammar induction. In *Proceedings of the EMNLP-CoNLL 2012*.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010a. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL-2010*.

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010b. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of ACL 2010*. Association for Computational Linguistics, Uppsala, Sweden, 1278–1287.

Wolfgang Tannebaum and Andreas Rauber. 2012. Acquiring lexical knowledge from query logs for query expansion in patent searching. In *Proceedings of ICSC'12*. IEEE Computer Society, Washington, DC, 336–338.

Gokhan Tur, Dilek Hakkani-Tur, Dustin Hillard, and Asli Celikyilmaz. 2011. Towards unsupervised spoken language understanding: Exploiting query click logs for slot filling. Annual Conference of the International Speech Communication Association (Interspeech).

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on EMNLP-CoNLL*. Association for Computational Linguistics, Prague, Czech Republic, 22–32.

Mo Yu, Tiejun Zhao, and Yalong Bai. 2013. Learning domain differences automatically for dependency parsing adaptation. In *IJCAI*, Francesca Rossi (Ed.). IJCAI/AAAI.

Received June 2015; revised September 2015; accepted March 2016