Combining Context Features by Canonical Belief Network for Chinese Part-Of-Speech Tagging

Hongzhi Xu and Chunping Li
School of Software, Tsinghua University
Key Laboratory for Information System Security, Ministry of Education China
xuhz05@mails.tsinghua.edu.cn
cli@tsinghua.edu.cn

Abstract

Part-Of-Speech(POS) tagging is the essential basis of Natural language processing(NLP). In this paper, we present an algorithm that combines a variety of context features, e.g. the POS tags of the words next to the word a that needs to be tagged and the context lexical information of a by Canonical Belief Network to together determine the POS tag of a. Experiments on a Chinese corpus are conducted to compare our algorithm with the standard HMM-based POS tagging and the POS tagging software ICTCLAS3.0. The experimental results show that our algorithm is more effective.

1 Introduction

Part-Of-Speech(POS) tagging is the essential basis of Natural language processing(NLP). It is the process in which each word is assigned to a corresponding POS tag that describes how this word be used in a sentence. Typically, the tags can be syntactic categories, such as noun, verb and so on. For Chinese language, word segmentation must be done before POS tagging, because, different from English sentences, there is no distinct boundary such as white space to separate different words(Sun, 2001). Also, Chinese word segmentation and POS tagging can be done at the same time(Ng, 2004)(Wang, 2006).

There are two main approaches for POS tagging: rule-based and statistical algorithms(Merialdo, 1994). Rule based POS tagging methods extratct rules from training corpus and use these rules to tag new sentences(Brill, 1992)(Brill, 1994). Statistic-based algorithms based on Belief Network(Murphy, 2001) such as Hidden-Markov-Model(HMM)(Cutting, 1992)(Thede, 1999), Lexicalized HMM(Lee, 2000) and Maximal-Entropy model(Ratnaparkhi, 1996) use the statistical information of a manually tagged corpus as background knowledge to tag new sentences. For example, the verb is mostly followed by a noun, an adverb or nothing, so if we are sure that a word a is a verb, we could say the word b following a has a large probability to be a noun. This could be helpful specially when b has a lot of possible POS tags or it is an unknown word.

Formally, this process relates to $Pr(noun|verb)$, $Pr(adverb|verb)$ and $Pr(nothing|verb)$, that can be estimated from the training corpus. HMM-based tagging is mainly based on such statistical information. Lexicalized HMM tagging not only considers the POS tags information to determine whether b is noun, adverb or nothing, but also considers the lexical information a itself. That is, it considers the probabilities $Pr(noun|a, verb)$, $Pr(adverb|a, verb)$ and $Pr(nothing|a, verb)$ for instance. Since combining more context information, Lexicalized HMM tagging gets a better performance(Lee, 2000).

The main problem of Lexicalized HMM is that it suffers from the data sparseness, so parameter smoothing is very important. In this paper, we present a new algorithm that combines several context information, e.g. the POS tags information and lexical information as features by Canonical Belief Network(Turtle, 1991) to together determine the tag

of a new word. The experiments show that our algorithm really performs well. Here, we don't explore Chinese word segmentation methods, and related information can be found in(Sun, 2001).

The rest of the paper is organized as follows. In section 2 and section 3, we describe the standard HMM-based tagging and Lexicalized HMM tagging respectively which are relevant to our algorithm. In section 4, we describe the Belief Network as a preliminary. In section 5, we present our algorithm that is based on Canonical Belief Network. Section 6 is the experiments and their results. In section 7, we have the conclusion and the future work.

2 Standard Hidden Markov Model

The problem of POS tagging can be formally defined as: given an observation(sentence) $w = \{w_1, w_2, ..., w_T\}$ and a POS tag set $TS = \{t_1, t_2, ..., t_M\}$, the task is to find a tag sequence $t = \{t_1, t_2, ..., t_T\}$, where $t_i \in TS$, that is the most possible one to explain the observation. That is to find t to maximize the probability $Pr(t|w)$. It can be rewritten by Bayesian rule as follows.

$$Pr(t|w) = \frac{Pr(w|t) \times Pr(t)}{Pr(w)}$$

As for any sequence t, the probability $Pr(w)$ is constant, we could ignore $Pr(w)$. For $Pr(t)$, it can be decomposed by the chain rule as follows.

$$\begin{aligned} Pr(t) &= Pr(t_1, t_2, ..., t_T) \\ &= Pr(t_1) \times Pr(t_2|t_1) \times Pr(t_3|t_1, t_2) \times \\ &\quad ... \times Pr(t_T|t_1, t_2, ..., t_{T-1}) \end{aligned}$$

Through this formula, we could find that the calculation is impossible because of the combination explosion of different POS tags. Generally, we use a n-gram especially $n = 2$ model to calculate Pr(t) approximately as follows.

$$\begin{aligned} Pr(t) = Pr(t_1|t_0) &\times Pr(t_2|t_1) \times Pr(t_3|t_2) \times \\ &... \times Pr(t_T|t_{T-1}) \end{aligned}$$

where t_0 is nothing. For $Pr(w|t)$, with an independent assumption, it can be calculated approximately as follows.

$$\begin{aligned} Pr(w|t) = Pr(w_1|t_1) &\times Pr(w_2|t_2) \times Pr(w_3|t_3) \\ &... \times Pr(w_T|t_T) \end{aligned}$$

Usually, the probability $Pr(t_i|t_{i-1})$ is called transition probability, and $Pr(w_i|t_i)$ is called the emission probability. They both can be estimated from the training set. This means that the tag t_i of word w_i is only determined by the tag t_{i-1} of word w_{i-1}. So, we could find the best sequence through a forward(left to right) process.

If we state all possible POS tags(stats) of each word and connect all possible t_{i-1} with all possible t_i and each edge is weighted by $Pr(t_i|t_{i-1})$, we could get a Directed Acyclic Graph(DAG). The searching process(decoding) that is involved in finding t that maximizes $Pr(t|w)$ can be explained as finding the path with the maximal probability. For this sub task, Viterbi is an efficient algorithm that can be used(Allen, 1995).

3 Lexicalized Hidden Markov Model

Lexicalized HMM is an improvement to the standard HMM. It substitutes the probability $Pr(t_i|t_{i-1})$ with $Pr(t_i|t_{i-J,i-1}, w_{i-L,i-1})$, and the probability $Pr(w_i|t_i)$ with $Pr(w_i|t_{i-K,i}, w_{i-I,i-1})$. In other words, the tag of word w_i is determined by the tags of the J words right before w_i and L words right before w_i. It uses more context information of w_i to determine its tag.

However, it will suffer from the data sparseness especially when the values of J, L, K and I are large, which means it needs an explosively larger training corpus to get a reliable estimation of these parameters, and smoothing techniques must be adopted to mitigate the problem. Back-off smoothing is used by Lexicalized HMM. In the back-off model, if a n-gram occurs more than k times in training corpus, then the estimation is used but discounted, or the estimation will use a shorter n-gram e.g. (n-1)-gram estimation as a back-off probability. So, it is a recursive process to estimate a n-gram parameter.

4 Belief Network

Belief Network is a probabilistic graphical model, which is also a DAG in which nodes represent random variables, and the arcs represent conditional independence assumptions. For example, the probability $Pr(A, B) = Pr(A) \times Pr(B|A)$ can be depicted as Figure 1(a), and if we decompose

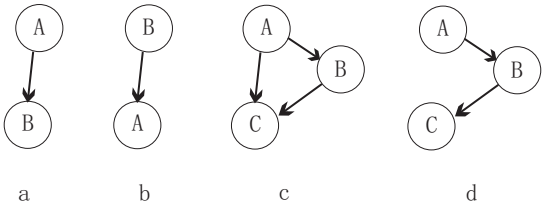

Figure 1: Some Belief Networks.

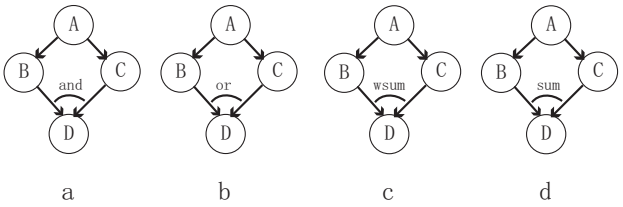

Figure 2: Canonical Belief Networks for $Pr(A, B, C, D)$.

$Pr(A, B) = Pr(B) \times Pr(A|B)$, it can be depicted as Figure 1(b). Similarly, the probability $Pr(A, B, C) = Pr(A) \times Pr(B|A) \times Pr(C|A, B)$ can be depicted as Figure 1(c).

As we have analyzed above, such decomposition would need us to estimate a large amount of parameters. In the belief network, a conditional independence relationship can be stated as follows: a node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes. For example, if we simplify the graph Figure 1(c) to graph Figure 1(d), it is equivalent to the decomposition: $Pr(A, B, C) = Pr(A) \times Pr(B|A) \times Pr(C|B)$, which is actually the same as that of HMM. More details about Belief Network can found in(Murphy, 2001).

5 Canonical Belief Network Based Part-Of-Speech Tagging

5.1 Canonical Belief Network

Canonical Belief Network was proposed by Turtle in 1991(Turtle, 1991), and it was used in information retrieval tasks. Four canonical forms are presented to combine different features, that is and, or, $wsum$ and sum to simplify the probability combination further. With the and relationship, it means that if a node in a DAG is true, then all of its parents must be true. With the or relationship, it means that if a node in a DAG is true, then at least one of its parents is true. With the $wsum$ relationship, it means that if a node in a DAG is true, it is determined by all of its parents and each parent has a different weight. With the sum relationship, it means that if a node in a DAG is true, it is determined by all of its parents and each parent has an equal weight.

For example, we want to evaluate the probability $Pr(D|A)$ or $Pr(D = true|A = true)$, and node D has two parents B and C, we could use the four canonical forms to evaluate $Pr(D|A)$ as shown in Figure 2. Suppose that $Pr(B|A) = p_1$ and $Pr(C|A) = p_2$, with the four canonical form and, or, $wsum$ and sum, we could get the following estimations respectively.

$$\begin{aligned} P_{and}(D|A) &= p_1 \times p_2 \\ P_{or}(D|A) &= 1 - (1 - p_1) \times (1 - p_2) \\ P_{wsum}(D|A) &= w_1 p_1 + w_2 p_2 \\ P_{sum}(D|A) &= (p_1 + p_2)/2 \end{aligned}$$

The standard Belief Network actually supposes that all the relationships are and. However, in real world, it is not the case. For example, we want to evaluate the probability that a person will use an umbrella, and there are two conditions that a person will use it: raining or a violent sunlight. If we use the standard Belief Network, it is impossible to display such situation, because it could not be raining and sunny at the same time. The or relationship could easily solve this problem.

5.2 Algorithm Description

Definition: A feature is defined as the context information of a tag/word, which can be POS tags, words or both. For example, $\{T_{i-J}, ..., T_{i-1}\}$ is a feature of tag t_i, $\{T_{i-J}, ..., T_i\}$ is a feature of word w_i, $\{T_{i-J}, ..., T_{i-1}, W_{i-L}, ..., W_{i-1}\}$ is a feature of tag t_i, $\{T_{i-K}, ..., T_i, W_{i-I}, ..., W_{i-1}\}$ is a feature of word w_i.

In our algorithm, we select 6 features for tag t_i, and select 2 features for word w_i, which are shown in Table 1. We can see that f_t^1, f_t^2 and f_t^3 are actually the n-gram features used in HMM, f_t^4, f_t^5 and f_t^6 are actually features used by lexicalized HMM.

We adopt the canonical form or to combine them as shown in Figure 3, and use the canonical form

	Features
t_i	f_t^1: $T_{i-3}, T_{i-2}, T_{i-1}$ f_t^2: T_{i-2}, T_{i-1} f_t^3: T_{i-1} f_t^4: $T_{i-3}, T_{i-2}, T_{i-1}, W_{i-3}, W_{i-2}, W_{i-1}$ f_t^5: $T_{i-2}, T_{i-1}, W_{i-2}, W_{i-1}$ f_t^6: T_{i-1}, W_{i-1}
w_i	f_w^1: T_{i-1}, T_i f_w^2: T_i

Table 1: Features used for t_i and w_i.

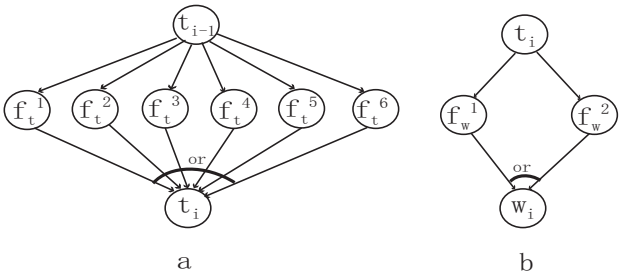

Figure 3: Canonical Belief Networks used in our algorithm.

and to combine features of t_i and w_i. Because we think that the POS tag of a new word can be determined if any one of the features can give a high confidence or implication of a certain POS tag. The probabilities $Pr(f_t^i|t_{i-1})$, $i = 1, ..., 6.$ are all 1, which means that all the features in the Canonical Belief Network are considered to estimate the tag t_i of word w_i when we have already estimated the tag t_{i-1} of word w_{i-1}. So, the transition probability could be calculated as follows.

$$p_{i-1,i}^{trans} = 1 - \prod_{j=1}^{6}[1 - Pr(t_i|f_t^j)]$$

In the same way, the probabilities $Pr(f_w^i|t_i)$, $i = 1, 2.$ are all 1. The emission probability could be calculated as follows.

$$p_i^{omit} = 1 - \prod_{j=1}^{2}[1 - Pr(w_i|f_w^j)]$$

Let's return to the POS tagging problem which needs to find a tag sequence t that maximizes the probability $Pr(t|w)$, given a word sequence w defined in Section 2. It is involved in evaluating two probabilities $Pr(t)$ and $Pr(w|t)$. With the Canonical Belief Network we just defined, they could be calculated as follows.

$$\begin{aligned} Pr(t) &= \textstyle\prod_{i=1}^{T} p_{i-1,i}^{trans} \\ Pr(w|t) &= \textstyle\prod_{i=1}^{T} p_i^{omit} \\ Pr(w,t) &= Pr(t) \times Pr(w|t) \end{aligned}$$

The canonical form *or* would not suffer from the data sparseness even though it refers to 4-gram, because if a 4-gram feature(f_t^1 for example) doesn't appear in the training corpus, the probability $Pr(t_i|f_t^1)$ is estimated as zero, which means the feature contributes nothing to determine the probability that word w_i gets a tag t_i, which is actually determined by a lower n-grams. Cases are the same for 3-gram, 2-gram and so on. In a special case, when a 4-gram (f_t^4 for example) appears in the training corpus and appears only once, the probability $Pr(t_i|f_t^1)$ will be 1, which means that the sentence or phrase we need to tag may have appeared in the training corpus, so we can tag the sentence or phrase with reference to the appeared sentence or phrase in the training corpus. This is an intuitional comprehension of our algorithm and its motivation.

Decoding: The problem of using high n-gram is the combination explosion especially for high grams. For example, consider the feature , suppose one word has 3 possible tags on average, then we have to evaluate $3^3 = 27$ cases for f_t^1, further, different features could get different combinations and the number of combinations will be $27^2 \times 9^2 \times 3^2 = 531441$. To solve the problem, we constrain all features to be consistent. For example, the tag t_{i-1} of feature f_t^1 must be same as that of feature f_t^2, f_t^3, f_t^4, f_t^5 and f_t^6 at one combination. The following features are not consistent, because the t_{i-1} in f_t^1 is VBP, while the t_{i-1} in f_t^4 is NN.

f_t^1 = JJ, NNS, VBP
f_t^4 = $JJ, NNS, NN, little, boys, book$

This will decrease the total combination to $3^3 = 27$. We use a greedy search scheme that is based on the classic decoding algorithm Viterbi. Suppose that the Viterbi algorithm has reached the state t_{i-1}, to calculate the best path from the start to t_i, we only use the tags on the best path from the start to t_{i-1} to calculate the probability. This decreases the total com-

bination to 3(the number of possible tags of t_{i-1}), which is the same as that of standard HMM.

6 Experiments

Dataset: We conduct our experiments on a Chinese corpus consisting of all news from January, 1998 of People's Daily, tagged with the tag set of Peking University(PKU), which contains 46 POS tags[1]. For the corpus, we randomly select 90% as the training set and the remaining 10% as the test set. The corpus information is shown in Table 2, where unknown words are the words that appear in test set but not in training set. The experiments are run on a machine with 2.4GHZ CPU, and 1GB memory.

	Training set	Test set
Words	1021592	112321
Sentences	163419	17777
Unknow words		2713

Table 2: Chinese corpus information.

Unknown Words: In our experiments, we first store all the words with their all possible POS tags in a dictionary. So, our algorithm gets all possible tags of a word through a dictionary. As for the word in the test set that doesn't appear in the training set, we give the probability $Pr(w_i|f_w^j)$ value 1, with all j. This processing is quite simple, however, it is enough to observe the relative performances of different POS taggers.

For Chinese word segmentation, we use the segmentation result of ICTCLAS3.0[2]. The segmentation result is shown in Table 3. *Sen-Prec* is the ratio of the sentences that are correctly segmented among all sentences in the test set.

Precision	Recall	F1	Sen-Prec
0.9811	0.9832	0.9822	0.9340

Table 3: Segmentation Result by ICTCLAS.

Open Test: We compare the POS tagging performance of our algorithm with the standard HMM, and ICTCLAS3.0. The experimental result is shown in Table 4. *Prec-Seg* is the POS tagging precision on the words that are correctly segmented. *Prec-Sen* is the ratio of the sentences that are correctly tagged among all sentences in the test set. *Prec-Sen-Seg* is the ratio of sentences that are correctly tagged among the sentences that are correctly segmented.

With the experiments, we can see that, our algorithm always gets the best performance. The ICTCLAS3.0 doesn't perform very well. However, this is probably because of that the tag set used by ICTCLAS3.0 is different from that of PKU. Even though it provides a mapping scheme from their tags to PKU tags, they may be not totally consistent. The published POS tagging precision of ICTCLAS3.0 is 94.63%, also our algorithm is a little better. This has proved that our algorithm is more effective for POS tagging task.

	ICTCLAS	HMM	CBN
Precision	0.9096	0.9388	**0.9465**
Recall	0.9115	0.9408	**0.9485**
F1	0.9105	0.9398	**0.9475**
Prec-Seg	0.9271	0.9569	**0.9647**
Prec-Sen	0.6342	0.7404	**0.7740**
Prec-Sen-Seg	0.6709	0.7927	**0.8287**

Table 4: Open test comparison result on Chinese corpus.

Close Test: As we have analyzed above in Section 5.2 that our algorithm takes advantage of more information in the training set. When a sentence or a phrase appears in the training set, it will help a lot to tag the new sentence correctly. To test whether this case really happens, we conduct a new experiment that is the same as the first one except that the test set is also added to the training set. The experimental result is shown in Table 5. We can see that the performance of our algorithm is greatly improved, while the HMM doesn't improve much, which further proves our analysis.

Even though our algorithm gives a satisfying performance, it may be able to be improved by adopting smoothing techniques to take advantage of more useful features, e.g. to make the probabilities such as $Pr(t_i|f_t^1)$, $Pr(t_i|f_t^2)$ not be zero. In addition, the adoption of techniques to deal with unknown words

[1] http://icl.pku.edu.cn/Introduction/corpustagging.htm

[2] ICTCLAS3.0 is a commercial software developed by Institute of Computing Technology, Chinese Academy of Science, that is used for Chinese word segmentation and POS tagging.

	ICTCLAS	HMM	CBN
Precision	0.9096	0.9407	**0.9658**
Recall	0.9115	0.9427	**0.9678**
F1	0.9105	0.9417	**0.9668**
Prec-Seg	0.9271	0.9588	**0.9843**
Prec-Sen	0.6342	0.7476	**0.8584**
Prec-Sen-Seg	0.6709	0.8004	**0.9191**

Table 5: Close test comparison result on Chinese corpus.

and techniques to combine with rules may also improve the performance of our algorithm. If we have a larger training corpus, it may be better to remove some confusing features such as f_t^3 and f_w^2, because they contain weak context information and this is why a higher n-gram model always performs better than a lower n-gram model when the training corpus is large enough. However, this should be validated further.

7 Conclusion and Future Work

In this paper, we present a novel algorithm that combines useful context features by Canonical Belief Network to together determine the tag of a new word. The 'or' node can allow us to use higher n-gram model although the training corpus may be not sufficient. In other words, it can overcome the data sparseness problem and make use of more information from the training corpus. We conduct experiments on a Chinese popular corpus to evaluate our algorithm, and the results have shown that it is powerful even in case that we don't deal with the unknown words and smooth the parameters.

We think that our algorithm could also be used for tagging English corpus. In addition, we only extract simple context information as features. We believe that there exists more useful features that can be used to improve our algorithm. For example, the syntax analysis could be combined as a new feature, because a POS sequence may be illegal even though it gets the maximal probability through our algorithm. Yet, these will be our future work.

Acknowledgement This work was supported by Chinese 973 Research Project under grant No. 2002CB312006.

References

Adwait Ratnaparkhi. 1996. *A Maximum Entropy Model for Part-Of-Speech Tagging*. In Proc. of the Empirical Methods in Natural Language Processing Conference(EMNLP'96), 133-142.

Bernard Merialdo. 1994. *Tagging English Text with a Probabilistic Model*. Computational Linguistics, 20(2):155–172.

Doug Cutting, Julian Kupied, Jan Pedersen and Penelope Sibun. 1992. *A Practical part-of-speech tagger*. In Proceedings of the 3rd Conference on Applied Natural Language Processing(ANLP'92), 133-140.

Eric Brill. 1992. *A simple rule-based part of speech tagger*. In Proc. of the 30th Conference on Applied Computational Linguistics(ACL'92), Trento, Italy, 112-116.

Eric Brill. 1994. *Some Advances in Transformation-Based Part of Speech Tagging*. In Proc. of the 12th National Conference on Artificial Intelligence(AAAI'94), 722-727.

Howard Turtle and W. Bruce Croft. 1991. *Evaluation of an Inference Network-Based Retrieval Model*. ACM Transactions on Information Systems, 9(3):187-222.

Hwee Tou Ng and Jin Kiat Low. 2004. *Chinese Part-of-Speech Tagging: One-at-a-Time or All-at-Once? Word-Based or Character-Based?*. In Proc. of the Empirical Methods in Natural Language Processing Conference(EMNLP'04).

James Allen. 1995. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company.

Kevin P. Murphy. 2001. *An introduction to graphical models*. Technical report, Intel Research Technical Report.

Maosong Sun and Jiayan Zou. 2001. *A critical appraisal of the research on Chinese word segmentation(In Chinese)*. Contemporary Linguistics, 3(1):22-32.

Mengqiu Wang and Yanxin Shi. 2006. *Using Part-of-Speech Reranking to Improve Chinese Word Segmentation*. In Proc. of the 5th SIGHAN Workshop on Chinese Language Processing, 205-208.

Sang-Zoo Lee, Jun-ichi Tsujii and Hae-Chang Rim. 2000. *Lexicalized Hidden Markov Models for Part-of-Speech Tagging*. In Proc. of 18th International Conference on Computational Linguistics(COLING'00), Saarbrucken, Germany, 481-487.

Scott M. Thede and Mary P. Harper. 1999. *A Second-Order Hidden Markov Model for Part-of-Speech Tagging*. In Proc. of the 37th Conference on Applied Computational Linguistics(ACL'99), 175-182.