Developing an Algorithm for Mining Semantics in Texts

Minhua Huang and Robert M. Haralick

Computer Science Department
The Graduate School and University Center
The City University of New York
New York, NY 10016

Abstract. This paper discusses an algorithm for identifying semantic arguments of a verb, word senses of a polysemous word, noun phrases in a sentence. The heart of the algorithm is a probabilistic graphical model. In contrast with other existed graphical models, such as Naive Bayes models, CRFs, HMMs, and MEMMs, this model determines a sequence of optimal class assignments among M choices for a sequence of N input symbols without using dynamic programming, running fast–$O(MN)$, and taking less memory space–$O(M)$. Experiments conducted on standard data sets show encourage results.
keywords. semantics, algorithm, text pattern, probabilistic graphical model, semantic argument, word sense, NP chunk

1 Introduction

Text patterns, such as semantic arguments of a verb, the meaning of a polysemous word, and noun phrases of a sentence, are essential patterns for capturing semantics in texts. For example, semantic arguments of a verb can be used to answer the questions of who, what, when, where, and why; the sense of a polysemous word can be used to understand the meaning of the word; noun phrases of a sentence can be used to extract the concepts of a sentence. Therefore, mining semantics patterns is useful.

In this paper, we discuss an algorithm for recognizing these text patterns. The heart of the algorithm is a probabilistic graphical model. In contrast with the existing graphical models, such as HMMs [1], MEMMs [2], or CRFs [3] that lead to an optimization for a sequence of class assignments to optimize the joint or conditional probability of the class assignments given a sequence of symbols using an implicit gain function where the gain is one if all class assignments are correct and zero if any assignment is wrong. Our model uses the gain function that gives partial credit for each correct assignment. With this criterion, the running time is reduced from $O(M^2N)$ to $O(MN)$ and the memory space is reduced from $O(MN)$ to $O(M)$, where M is the cardinality of the set of class assignments and N is the length of an input symbol sequence. Moreover, by applying the method on standard data sets for recognizing three patterns, we find that our results exceed or approach the current state of the art.

2 Creating the Model

2.1 Economic Gain Function

Let $s =< s_1, \ldots, s_N >$ be a sequence of N symbols. Let C be a set of M classes, $C = \{C_1, \ldots, C_M\}$. Let $c =< c_1, ..., c_N >$ be a sequence of assigned classes. Let $c^T =< c_1^T, ..., c_N^T >$ be a sequence of true classes. Let e be the economic gain function $e : C^N \times C^N \rightarrow \mathcal{R}^+$. For the existing probabilistic graphical models, such as HMMs [1], MEMMs [2], or CRFs [3], the economic gain function $e(c_1^T, ..., c_N^T, c_1, ..., c_N) = 1$ when $c_1^T, ..., c_N^T = c_1, ..., c_N$ and $e(c_1^T, ..., c_N^T, c_1, ..., c_N) = 0$ otherwise. That is, the gain function specifies a gain of one if all the class assignments are correct and zero if one or more of the class

assignments are wrong. No partial credit is given for some correct assignments. As a consequence, entire wrong classification subsequences can be produced around an incorrectly assigned symbol just because it has been subjected to random noise or perturbations. We use a gain function that gains some value for each correct assignment. It is defined by:

$$e(c_1^T, ..., c_N^T, c_1, ..., c_N) = \sum_{n=1}^{N} e(c_n, c_n^T) \tag{1}$$

$$where: \quad e(c_n^T, c_n) = \begin{cases} > 0, & when\ c_n^T = c_n \\ 0, & otherwise \end{cases}$$

Compared with the previous gain function implicitly employed by the HMMs, MEMMs, and CRRFs, our gain function gives partial credits for correct assignments. In fact, this gain function is also implicitly employed by the context independent Bayes model where each symbol class pair is independent of all the other symbol class pairs. To maximize the expected gain under equation (1), we have:

$$E(e) = \sum_{n=1}^{N} \sum_{c_n^T} e(c_n^T, c_n) p(c_n^T, s_1, .., s_N)$$

When the gain matrix is diagonal and positive, then:

$$E(e) = \sum_{n=1}^{N} e(c_n, c_n) p(c_n, s_1, ..., s_N)$$

When the gain matrix is the identity, assigning the value 1 for a correct assignment and the value 0 for an incorrect assignment, then:

$$E(e) = \sum_{n=1}^{N} p(c_n, s_1, ..., s_N)$$

In this case, maximizing the expected gain is associated with maximizing $p(c_n|s_1, ..., s_N)$, where $n = 1, ..., N$.

$$max(E(e)) = \sum_{n=1}^{N} \underset{c_n \in C}{max}\, p(c_n, s_1, ..., s_N)$$

2.2 Building The Model

To find the assigned class sequence $< c_1, ..., c_N >$ for the input sequence $< s_1, .., s_N >$ that maximizes the expected gain, we only need to find an assigned class $c_n, n = 1, ..., N$ that maximizes the joint probability function $p(c_n, s_1, ..., s_N)$. This leads to the mathematical representation for the probability in Equation (2).

$$p(c_1, \ldots, c_N|s_1, \ldots, s_N) = \frac{\prod_{n=1}^{N} p(s_{n-1}|s_n, c_n) p(s_{n+1}|s_n, c_n) p(s_n|c_n) p(c_n)}{\sum_{c_k \in C} \prod_{k=1}^{N} p(s_{k-1}|s_k, c_k) p(s_{k+1}|s_k, c_k) p(s_k|c_k) p(c_k)} \tag{2}$$

2.3 Finding $< c_1^*, \ldots, c_N^* >$

By Equation (2), to find a class sequence $< c_1^*, \ldots, c_N^* >$ given a sequence of symbols $< s_1, \ldots, s_N >$, we only need to find c_n^* for s_n individually. Note, the denominator in (2) is a constant. Therefore, it does not effect a decision for assigning c_i to s_i.

$$< c_1^*, c_2^*, ..., c_N^* >= \prod_{n=1}^{N} \underset{c_n \in C}{argmax}\, p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n) \tag{3}$$

Letting $f(s_{n-1}, s_n, s_{n+1}, c_n) = p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$

$$< c_1^*, c_2^*, ..., c_N^* >= \prod_{n=1}^{N} \underset{c_n \in C}{argmax}\, f(s_{n-1}, s_n, s_{n+1}, c_n)$$

2.4 Complexities

HMMs or CRFs employ dynamic programming to obtain a sequence optimal of classes for a sequence of symbols by computing a joint probability $p(s_1 \ldots s_n\ c_1 \ldots c_N)$ or a conditional probability $p(c_1 \ldots c_N|\ s_1 \ldots s_n)$. By dynamic programming, an optimal class for the current symbol is obtained based on an optimal class of the previous symbol. Therefore, the optimal class for the last symbol is determined after the last symbol has been reached. The optimal class sequence needs to be determined by tracing back from the last optimal class to the first optimal class. For each symbol, information for M classes needs to be stored. Hence, for a sequence of N symbols, we need to have $O(M^2N)$ time complexity and $O(M * N)$ memory complexity. For our model, by equation (3), for each symbol s_n, we need to assign a c_n, such that

$$f(s_{n-1}, s_n, s_{n+1}, c_n) \geq f(s_{n-1}, s_n, s_{n+1}, c_n'),\ \ c_n' \in C$$

Time Complexity. To compute $f(s_{n-1}, s_n, s_{n+1}, c_n)$, we need to have four multiplications. To obtain the maximum probability value we require $M-1$ comparisons. In the case of a sequence of N symbols, we need

$$T_c = 4 * N * (M-1) * (L-1) = O(N * M * L)$$

Memory Complexity. Because the global maximum probability is determined by each local maximal probability, for a path of N symbols, we only need to store the information of the current node. That is, we need only store M probability values in order to find the maximal probability value. Therefore,

$$M_c = M = O(M)$$

Comparisons We compute ratios of time complexity and memory complexity of our model to HMMs and CRFs to see differences. By observing these two ratios, we see that if we need to recognize a sequence of N symbols with M categories, our model only take $\frac{1}{M}$ time and $\frac{1}{N}$ memory space of HMMs or CRFs. For example, if the cardinality of C is $(M = 8)$, for a sequence of sixteen symbols $(N = 30)$, our method only needs to have $\frac{1}{8}$ time and $\frac{1}{30}$ memory space of a HMM or a CRF to

recognize this sequence.
Ratio of Time Complexity.

$$\frac{NM}{M^2N} = \frac{1}{M}$$

Ratio of Memory Complexity.

$$\frac{M}{M * N} = \frac{1}{N}$$

3 Three tasks

3.1 Identifying Semantic Arguments of a Verb

Let $T = (V, E, r, A, L)$ be a labeled rooted tree associated with a sentence, where V is a set of vertices, E is a set of edges, $E \subseteq V \times V$, r is the root, A is an alphabet defined by [4], and L is a labeling function $L : V \rightarrow A$ that assigns labels to vertices. The parse tree of the sentence takes the form of T. Let π be a set of labels, s.t. $\pi \subseteq A$. Let $C = \{C_1, C_2\}$ be a set of classes, where C_1 represents that a path will be extended from the current node to an adjacent node; C_2 represents that a path will not be extended from the current node to an adjacent node.

The Procedure

- Form a path $\mathcal{P}(x) = \tau_1, \rightarrow \ldots, \rightarrow \tau_K$, $x \in V$, $L(x) \in \pi$, and x is not a node in $\mathcal{P}'(y)$, $\mathcal{P}'(y)$ is a path that has been already formed previously.

$$< \tau_1, \ldots, \tau_K >= \underset{b_1,\ldots,b_K}{argmax}\, p(c_1, \ldots, c_K, b_1, \ldots, b_K)$$

  Note, $c_k \in C$, $b_k \in V$, $b_{k-1}b_k \in E$.
- Form a set of roots $R(x) = \{r_i | i = 1 \ldots M\}$ from $\mathcal{P}(x)$, where $r_i \leq \tau_k$.
  - For all siblings of τ_k, find z, s.t. $L(z) \notin \pi$ and $z \notin \{\tau_k | k = 1, \ldots, K\}$, then $R(x) \leftarrow R(x) \cup \{z\}$
  - For all children of τ_k, find y, s.t. $L(y) \notin \pi$ *and* $y \notin \{\tau_k | k = 1, \ldots, K\}$, then $R(x) \leftarrow R(x) \cup \{y\}$
- Find a rooted forest $F(x) = \{T_i | i \in \{1, .., I\}\}$,
  - Each T_i is induced from the root r_i by all its co-dependents.
  - For each $T_i \in F(x)$, the leaves $\{l_i^1, ..., l_i^K\}$ correspond to one of the semantic arguments of x.

3.2 Identifying the Sense of a Word

Let $S =< s_1, .., s_t, .., s_N >$ be a sequence of symbols associated with a sentence, $s_t \in S$ be a given ambiguous symbol that needs to be disambiguated . Let $C = \{C_m | m = 1, ..., M\}$, C be a set of predefined senses of the ambiguous symbol s_t.

The Contexts The Context of an ambiguous symbol s_t is a $k-tuple$, represented by T_t. Each element in T_t is a symbol in S, $T_t = (t_1, ..., t_K)$, $t_k \in S$, and $K \leq N$.

The Procedure

- Find the context T_t for s_t.
- Find a sequence of classes $< c_1^*, \ldots, c_K^* >$ for $T_t = (t_1, ..., t_K)$, s.t.

$$< c_1^*, \ldots, c_K^* >= \underset{c_1,\ldots,c_K}{argmax}\, p(c_1, \ldots, c_K | t_1, \ldots, t_K)$$

- Assign C_j to s_t if and only if

$$\#\{k \mid c_k = C_j\} \geq \#\{k \mid c_k = C_m\},\ m = 1, \ldots, M$$

3.3 Identifying Noun Phrases

Let S be a sequence of symbols associated with a sentence, $S =< s_1, ..., s_i, ..., s_N >$, where s_i is the pair (word, its speech tag). Let C be a set of classes, $C = \{C_1, C_2, C_3\}$, where C_1 represents that a symbol is inside a noun phrase, C_2 represents that a symbol is not in a noun phrase, and C_3 represents that a symbol starts at a new noun phrase.

Building Blocks $\mathcal{B}$ is a block if and only if:

1. For some $i \leq j$, $\mathcal{B} = < (s_i, c_i), (s_{i+1}, c_{i+1}), \ldots, (s_j, c_j) >$
2. $c_i \in \{C_1, C_3\}$
3. $c_n = C_1$, $n = i + 1, \ldots, j$
4. For some $\mathcal{B}'$, if $\mathcal{B}' \supseteq \mathcal{B}$ and $\mathcal{B}'$ satisfies 1, 2, 3 then $\mathcal{B}' \subseteq \mathcal{B}$

The Procedure

- Find a sequence of classes $< c_1^*, \ldots, c_N^* >$, s.t.

$$< c_1^*, \ldots, c_N^* >= \underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N | s_1, \ldots, c_N)$$

- Find $\{B_1, \ldots, B_M\}$, where each B_m is a block satisfying the definition of $\mathcal{B}$.

4 Evaluation

In order to evaluate our method, we have conducted two types of tests. In the first type of tests, we test our method on three tasks on standard data sets and compare the results published by other researchers on the same data sets. In the second type of tests, we implement the context independent Naive Bayes method and test it on the selected tasks and compare the results with our method.

4.1 Experiments Set Up

Data sets. Data sets that we have selected for our method are WSJ data from the Penn TreeBank and the PropBank [4], data developed by [5] [6], and WSJ data from the Penn TreeBank [7] and CoNLL-2000 Shared Task [8]. Our reasons for using these data sets were that they have been studied by numbers of other researchers and many results have been published over the years.

Evaluation metrics. The evaluation metrics designed for testing the first and the third tasks were *precision , recall, f-measure* (F_1) and for testing the second task were *accuracy*. The reason of selecting different evaluation methods was based on the design of classes described in sections 3.1, 3.2, and 3.3 [1]. One of the classes was not needed to be evaluated in task one and three while all classes were needed to be evaluated in task two.

Training set and testing set distributions. We have used a 10-fold cross validation technique for obtaining our result for all experiments.

4.2 The First Type of Test

First Task Results. The data set, the section 00 of WSJ from Penn Treebank and PropBank [4], was used for testing in the first task. A total of 223 sentences is in files *20, 37, 49, and 89*. Associated with each of these sentences, it was an automatically determined parse tree provided by Penn Treebank. These parse trees had an average accuracy of 95.0%. Among these sentences, there were 621 verbs. Each verb had an average of three semantic arguments. Hence about 2000 semantic arguments were used. The semantic arguments were provided by PropBank. These were created manually.

Among 621 verbs, about 560 verbs were used for obtaining probability values while about 60 verbs were used to form paths based on these probability values. Some of the paths were listed on Figure 1. They were obtained based on the procedure described in Section 3.3 by applying 10-fold cross validation technique. We noticed that 86% paths fell into the first three patterns in Figure 1.

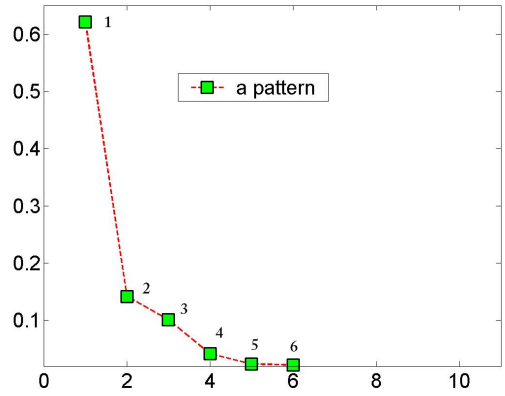

Fig. 1. Six patterns of paths: 1.$VBZ(VBD, VBG, VBP, VBN, VB) \rightarrow VP$, 2.$MD(TO) \rightarrow VP \rightarrow VP \rightarrow VB$, 3.$VBP(VBZ, VBD) \rightarrow VP \rightarrow VP \rightarrow VBN$, 4.$VBD(VBZ, VBN) \rightarrow VP \rightarrow RB \rightarrow VP \rightarrow VB$, 5.$TO \rightarrow VP \rightarrow VP \rightarrow VB \rightarrow VP \rightarrow VBN$, 6.$MD \rightarrow VP \rightarrow RB \rightarrow VP \rightarrow VBP(VB) \rightarrow VP \rightarrow VBN$.

After forming a path for a verb in the test instances, a set of roots were found. From these roots, a set of labeled rooted subtrees, whose leaves were associating with semantic arguments of the verb,

[1] There was no class that represented none of these classes in the first task and third task while every class was a distinct sense in the second task.

was formed. The test results were shown in Table 1. On the average, each time among $\frac{1}{10}$ of the semantic arguments were classified, about 93% semantic arguments were correctly identified and 7% semantic arguments were mistakenly identified. By checking these classified instances, we found that our method was very effective in the case of a semantic argument being a sequence of consecutive words. However, if a semantic argument consisted of two or more word fragments, separated by some phrases, our algorithm was less effective. The reason was that these phrases were parts of leaves of a tree induced from a root determined by our algorithm. This suggests that in order to exclude phrases from a semantic argument, we need to develop a method so that a set of subroots needs to be found. Each of them corresponds to a fragment of a semantic argument. Moreover, other misclassified instances were generated by errors carried in original syntactic trees.

Table 1. The First task on WSJ data

Files 20, 37, 49, 89	Precision %	Recall %	F-Measure %
F-measure	92.335	94.1675	93.2512
Std	0.6195	0.5174	0.4605

Second Task Results. We tested our method for identifying the sense of a word on the data sets *line, hard, serve*, and *interest*. The senses' descriptions and instances' distributions could be found in [5] and [6]. In these data sets, *line* and *interest* were polysemous nouns, *hard* was a polysemous adjective, and *serve* was a polysemous verb. In our experiment, *line* had 6 senses, *serve* had 4 senses, *hard* had 3 senses, and *interest* had 3 senses (3 other senses were omitted due to insufficient number of instances). The test metric that we have used was *accuracy*.

We formed the context of each given target word by including the left four open class words and the right four open class words combining with the left word and the right word for each of these words. The test results were shown in Table 2.

Table 2. The second task on *line-serve-hard-interest* data

Ambiguous word	Senses	Accuracy %	Standard deviation %
line (noun)	6	81.16	1.92
	3	85.25	2.13
serve (verb)	4	79.80	1.90
hard (adjective)	3	82.88	3.10
interest (noun)	3	92.10	2.21

We found that misclassified instances were primarily generated by the ambiguity of context words. For example in Table 2, comparing with the three sense noun *interest* and the three sense noun *line* obtained by selecting three senses at each time from six senses and examining all twenty combinations, we found that the accuracy of the word *interest* was almost 9% higher than the accuracy for the word *line*. Moreover, by examining accuracies generated from each combination for the word *line*, we found that some combination had the highest average accuracy, for instance $S_1S_2S_4$ had an average accuracy

of 91.7% while some combination had the lowest average accuracy, for instance $S_1S_3S_5$ had an average accuracy of 77.1%. The difference was almost 20%. By carefully checking these misclassified instances, We learned that if two senses were similar to each other, there were more chances that their contexts consisted of the same words. As a consequence, the misclassification rate increases.

Moreover, by observing the outputs of two polysemous nouns *line* and *interest*, we found that as the number of senses of a polysemous noun increasing, the accuracy decreased. This suggested that nouns with a larger number of senses were more difficult to recognize than nouns with small number of senses by our algorithm. Furthermore, by comparing accuracies, we noticed that nouns were relatively easier to identify than adjectives or verbs. From comparing the standard deviations we noticed that accuracies generated by our algorithm on adjectives had a larger variance than that on nouns or verbs.

Comparisons Our results are better than the results reported by other WSD researchers [9] and [5]. Our method achieves an average accuracy of 81.12% for identifying the six sense noun *line* using 2450 training context words while the method proposed by [5] achieves the average accuracy 73% using 8900 training context words. Moreover, an experiment using the Latent Semantic Analysis method conducted by [9] achieves an average accuracy of 75% for identifying only three senses of *line*. The comparisons are shown in Table 3.

Table 3. Comparisons on recognizing word sense on **line** data

Method	Bayesian [5]	The algorithm	LSA [9]	Context vector [5]	Neural Network [5]
Accuracy	71	81	75	72	76

Third Task Results. Three types of symbols were designed for identifying NP chunks on CoNLL-2000 Shared Task data set. They were the lexicon of a word, the POS tag of a word, and the lexicon and the part of speech (POS) tag of a word. The results were shown in the second row of Table 4. By comparing the results, we noticed that if the model was built only on the lexical information, it had the lowest performance 89.75%. The model's performance improved 3% if it was constructed by POS tags. The model achieved the best performance of 95.59% if both lexicon and POS tags were included.

Different from the first experiment, the second experiment on the WSJ data from Penn Treebank used only one type of symbol: the lexicon and the POS tag of a word. The main reason for using this data set was that we wanted to see whether the performance of our model could be improved when it was built on more data. In this case, the training set was seven times larger than the CoNLL-2000 shared task training data set. The test results were shown in the third row of Table 3. Note, data inside parentheses in the table represented standard deviation.

Compared with the results on these two data sets, we noticed that the average precision was improved about 2.7% from 95.15% to 97.73% . The average recall was improved about 2.8% from 96.05% to 98.65%. The average F-measure was improved about 2.7% from 95.59% to 98.2% as the training sets expanded to seven times larger. This suggested that the larger the training sets, the better the results.

Comparisons Table 5 shows the best performances of the related methods [1] [10] [11] [12] [13] [14] on the CoNLL-2000 shared task data. Among of these methods, the role based learning method achieves the worst F-measure performance and our method achieves the best F-measure performance.

Table 4. The test results on the CoNLL-2000 and WSJ data

Data	Symbol type	Precision %	Recall %	F-measure %
CoNLL-2000	Lexicon + POS	95.15	96.05	95.59
	POS	92.27	93.76	92.76
	Lexicon	86.27	93.35	89.75
WSJ	Lexicon + POS	97.73	98.65	98.18
		(0.19)	(0.14)	(0.08)

Table 5. Comparisons for different methods on the CoNLL-2000 data set

Method	RBL [13]	HMM [1]	NB [2]	MEMM [10]	VP [11]	CRF [10]	SVM [12]	the algorithm
F-measure	91.54	93.52	93.69	93.70	93.74	94.38	94.45	95.74

4.3 The Second Type of Test

The Context Independent Bayes Model We implemented the context independent Bayes model represented by $p(c_1, \ldots, c_N|s_1, \ldots, s_N) = \prod_{i=1}^{N} p(s_i|c_i)$. As mentioned in Section 2.1, this model also implicitly employed the economic gain function where each symbol class pair was independent of all the other symbol class pairs.

In order to compare with the context independent Bayes model with our method, we conducted experiments on two date sets for two tasks: CONLL-2000 data set for identifying noun phrases in a sentence and *interest* data for identifying the sense of the word *interest*. We still used $10 - folder$ cross validation technique to obtain an average.

Figure 2 and Figure 3 showed the results. In these figures, $f - measure_2/accuracy_2$ represented an F-measure/accuracy obtained by our method while $f - measure_1/accuracy_1$ represented an F-measure/accuracy obtained by the context independent Bayes model. In each case, we put all results generated by these two methods into a pool, and randomly selected a pair of results (each component in the pair was generated by different methods) and computed their difference. We ran this procedure for five thousand times.

In Figure 2, the left side of the figure represented the number of occurrences that an F-measure obtained by our method was lesser than an F-measure obtained by the context independent Naive Bayes while the right side of the figure represented the number of occurrences that an F-measure obtained by our method was greater that an F-measure obtained by the context independent Naive Bayes.

By using $\frac{mean\{f-measure_2 - f-measure_1\}}{mean\{f-measure_1\}}$, we found that our method achieved a 2.24% better average F-measure than the context independent Bayes model on identifying NP chunks with the confidence of 90.26%. In the same way, by observing Figure 3, we found that our method achieved a 5.44% better average accuracy than the context independent Bayes model on identifying the sense of the word *interest* with the confidence of 96.98%.

4.4 Discussion

Different Graphical Representations. Currently existing graphical models used by most researchers are HMMs [2] [1], MEMMs[2], and CRFs[3] [10]. These models are built to obtain an optimal sequence of N classes $c =< c_1, \ldots, c_N >$ from a sequence of N symbols $s =< s_1, \ldots, s_N >$ by finding

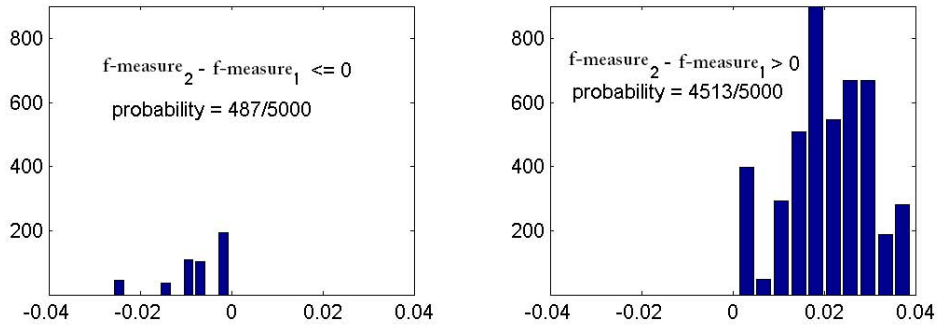

Fig. 2. Comparisons of Context Independent Bayes with our method on the CoNLL-2000 data set.

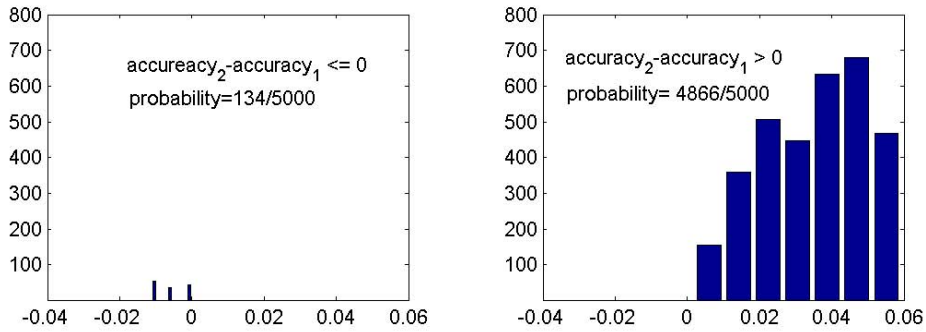

Fig. 3. Comparisons of Context Independent Bayes with our method on *interest* data set.

the maximum value of the joint probability $p(c, s)$ or the conditional probability $p(c|s)$. These graphical models are shown in Figure 4. While HMMs and MEMMs are directed graphical models, CRFs and our model are undirected graphical models. While others have a link from c_{i-1} to c_i, our model links c_i and s_{i+1} and links c_i and s_{i-1}. We believe that c_i can be better predicated from s_{i-1} and s_{i+1} rather than c_{i-1} when symbols contain several types of information. For example, in the case of NP chunking, POS tag information carried on a symbol is much useful than the class information assigned on the previous symbol.

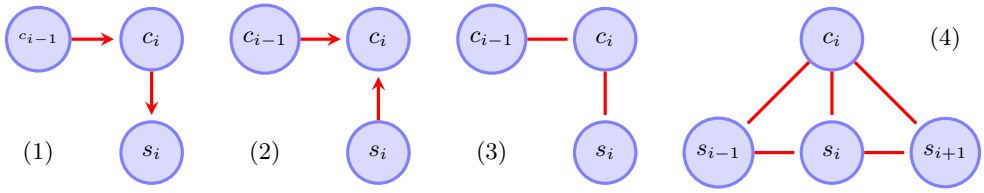

Fig. 4. (1): a HMM model, (3): a MEMM model, (4): a CRF model, and (5): the model presented by this paper

Different Assumptions. In the conditional independence graph, $s_i,\ i = 1, ..., N$ and $c_i,\ i = 1, ..., N$ are nodes. We have $2N$ nodes in total. If no assumption is made, there is a link between every pair of nodes. The degree of each node should be $2N - 1$. Some assumptions are made for the graphical models shown in Figure 4. Compared with these graphic models, we find that for each s_i, the degree of our model is three while others are two, which indicates that our model has less assumptions. The HMM model is built under two conditional independence assumptions. First, given its previous class, the current class is independent of other classes. Moreover, given its current class, the symbol is independent of other classes and symbols. The $MEMM$ model is built under one conditional independence assumption. Given its previous class and the current symbol, the current class is independent of other classes and symbols. The CRF model is built under the same two conditional assumptions as the HMM model. The model presented in this paper makes one conditional independence assumption. Given the current, the preceding, and the succeeding symbol, the current class is independent of other classes and symbols.

Comparisons Related To The Three Tasks. A number of methods for NP chunking [15] [16] [1] [17] [12], word sense disambiguation [18] [19] [5] [20] [21], and semantic role labelling [22] [23] [24] [25] [26] have been developed over the years. We adopted some ideas from these methods. For instance, in NP chunking, we follow Ramshaw's idea [16] of designing three categories for a word in a sentence to determine whether the word is inside a NP chunk, outside a NP chunk, or should start a new NP chunk. However, most methods for this task use HMMs [2] [1], MEMMs[2], and CRFs[3] [10]. In contrast with these methods, we created a new algorithm for these tasks. The core technique of the method is a probabilistic graphical model. This model is fast, uses less memory, and works well for text data.

In the WSD task, in contrast with other WSD methods, the polysemous word is represented by a sequence of context symbols, each symbol is a ordered pair of the lexicon and the POS tag of a word. Each symbol is represented by it's left symbol and right symbol. Moreover, in the semantic argument identification task, most existing methods transform a syntactic tree into a sequence of constituents. Each argument of a verb is represented by a set of constituents. Each constituent is represented by a set of features. These features are extracted based on linguistic knowledge and local knowledge of the tree structure. Finally, sophisticated classifiers such as support vector machines or maximum entropy modeling classifiers are employed to identify semantic arguments of each verb. In contrast to these methods, our method is based on the idea that if a sentence has a correspondent labeled rooted tree, a semantic argument of a verb in the sentence will be associated with a labeled rooted subtree. Hence, all semantic arguments of a verb in the sentence will be represented by a set of labeled rooted subtrees. For each verb node v, there exists a path, from which, all roots of the subtrees will be extracted. Obviously, the unique feature, which is a path, represents all semantic arguments of a verb. We find such a path for each verb in a labeled rooted tree associated with a sentence by the probabilistic graphic model.

5 Conclusions

We developed an algorithm for identifying three types of semantic patterns: the semantic arguments of a verb, the sense of an ambiguous word, and the noun phrases of a sentence, in texts based on a probabilistic graphical model. By this model, a sequence of optimal classes (or a path) for a sequence of symbols (or nodes) is obtained in a simple way - no need for dynamic programming, fast - O(NM), and less memory spaces - O(M) compared with other existing models such as HMMs, CRFs, and $MEMM$s. Moreover, because the global maximum probability is achieved by finding assignments that maximize the local probabilities, where the local probabilities take into account neighboring symbol adjacency, and the method provides credit for each correct answer, our performance is comparable or better than other published results on the same data sets.

References

1. Molina, A., Pla, F., tics, D.D.S.I., Hammerton, J., Osborne, M., Armstrong, S., Daelemans, W.: Shallow parsing using specialized hmms. Journal of Machine Learning Research **2** (2002) 595–613
2. MaCallum, A., Freitag, D., Pereira, F.: Maximum entropy markov models for information extraction and segmentation. In: Proceedings of 17th International Conf. on Machine Learning. (2000) 591–598
3. Lafferty, J., MaCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of 18th International Conf. on Machine Learning. (2001) 282–289
4. Weischedel, R., Palmer, M., Marcus, M., Hovy, E.: Ontonotes release 2.0 with ontonotes db tool v. 0.92 beta and ontoviewer v.0.9 beta. In: http://www.bbn.com/NLP/OntoNotes. (2007)
5. Leacock, C., Towell, G., Voorhees, E.: Corpus based statistical sense resolution. In: Proceedings of the workshop on Human Language Technology. (1993) 260 – 265
6. Bruce, R., Wiebe, J.: Word-sense disambiguation using decomposable models. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. (1994) 139–146
7. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics **19**(2) (1994) 313–330
8. Tjong, E.F., Sang, K.: Introduction to the conll-2000 shared task: Chunking. In: Proceedings of CoNLL-2000. (2000) 127–132
9. Levin, E., Sharifi, M., Ball, J.: Evaluation of utility of lsa for word sense discrimination. In: Preceedings of HLT-NAACL. (2006) 77 – 80
10. Sha, F., Fereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL. (2003) 213–220
11. Carreras, X., Mrquez, L.: Phrase recognition by filtering and ranking with perceptrons. In: the International Conference on Recent Advances on Natural Language Processing. (2003)
12. Wu-Chieh, Wu, Lee, Y.S., Yang, J.C.: Robust and efficient multiclass svm models for phrase pattern recognition. Pattern Recognition **41** (2008) 2874–2889
13. Veenstra, J., den BoschJ, A.V.: Single-classifier memory-based phrase chunking. In: Preceedings of CoNLL-2000 and LLL-2000. (2000) 157–159
14. Huang, M., Haralick, R.M.: Recognizing Patterns in Texts. River (2010)
15. Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the second conference on Applied natural language processing. (1988) 136 – 143
16. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Proceedings of the Third Workshop on Very Large Corpora. (1995) 82–94
17. Abney, S., Abney, S.P.: Parsing by chunks. In: Principle-Based Parsing, Kluwer Academic Publishers (1991) 257–278
18. Hearst, M.A.: Noun homograph disambiguation using local context in large text corpora. In: Proceedings of the Seventh Annual Conference of the UW centre for the New OED and Text Research. (1991) 1–22
19. Gale, W., Church, K., Yarowsky, D.: A method for disambiguating word senses in a large corpus. In: Computers and the Humanities. (1992) 415–439
20. Leacock, C., Miller, G.A., Chodorow, M.: Using corpus statistics and wordnet relations for sense identification. Computational Linguist. **24** (1998) 147–165
21. Yarowsky, D.: Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and frech. In: Preceedings of the 32nd Annual Meeting. (1994)
22. Gildea, D., Jurafsky, D.: Automatic labelling of semantic roles. Computational Linguistics (2002) 245–288
23. Baldewein, U., Erk, K., Pad, S., Prescher, D.: Semantic role labeling with chunk sequences. In: Proceedings of CoNLL-2004 Shared Task. (2004)
24. Cohn, T., Blunsom, P.: Semantic role labelling with tree conditional random fields. In: Proceedings of CoNLL-2005 Shared Task. (2005)
25. Hacioglu, K.: A semantic chunking model based on tagging. In: Proceedings of HLT/NACCL-2004. (2004)
26. Hacioglu, K.: Semantic role labeling using dependency trees. In: Proceedings of Coling 2004, Geneva, Switzerland, COLING (Aug 23–Aug 27 2004) 1273–1276