Contents lists available at ScienceDirect

Pattern Recognition Letters

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

A method for discovering knowledge in texts

Minhua Huang[a,*], Robert M. Haralick[b]

[a] Computer Science, Texas A&M University – Corpus Christi, 6300 Ocean Dr, Corpus Christi, TX 78412, USA
[b] Computer Science, Graduate Center of New York City University, NY 10016, USA

ARTICLE INFO

Article history:
Available online 16 March 2018

Keywords:
Probabilistic graphic models
Cliques
Separators
NP chunks
Word senses
Semantic arguments

ABSTRACT

We discuss a more powerful probabilistic graphical model for discovering semantic patterns from sequential text data, such as sentences. It is developed based on the idea that each word (or each symbol) in a sentence itself might carry lexical, semantic, or syntactic information, which can be used to replace conditional dependences in existing methods. Hence, our method has fewer conditional independence assumptions in contrast to these existing probabilistic graphical methods, such as CRFs, HMMs, MEMMs and Naive Bayes. Moreover, our method does not need to employ dynamic programming and therefore the on-line time complexity and memory complexity are reduced. We test the method on discovering noun phrases, the meaning of an ambiguous word, and semantic arguments of a verb in a sentence. We find that the misclassification rate is smaller compared to previously published results on the same data sets. For example, the method achieves an average *f*-measure of 98.25% for recognizing noun phrases on WSJ data from Penn Treebank; an average accuracy of 81.12% for recognizing the six sense word *line*; an average *f*-measure of 93.61% for classifying semantic argument boundaries of a verb in a sentence on WSJ data from Penn Treebank and PropBank.

1. Introduction

Recently, Neural Networks (NNs), such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown great success in some NLP tasks [36]. The most commonly used type of RNNs in NLP is Long-Short Term Memory (LSTM) [13] in which three gates are designed for capturing dependencies. However, these networks are designed for handling real-valued vectors rather than for handing symbols. In fact, the natural language is a sequence of symbols. In texts, various of resources of information are existed among symbols. For example, there are syntactic relations, lexical relations, and semantic relations among words in a sentence. The original resources of information might be difficult to carry when a word is represented by a real-valued vector with a predetermined size. Moreover, the running time complexity would be increased and errors could be created by transforming a symbol into a vector, and then a result vector back to a symbol. Besides, the vector representations are more difficult to interpret and manipulate than symbol representations. For example, it is difficult to know and fix an error in a vector representation. Furthermore, some tasks, such as inference and decision making are ineffective on these networks [25]. In this paper, we present a probabilistic graphical model built based on the nature of language data. Because of our model is built on symbol data, the properties of natural language are remained. Our model runs faster and is easy to interpret and manipulate. Our model can perform sequential decision making.

Probabilistic graphical models, such as Hidden Markov models (HMMs) [19,28], Maximum-entropy Markov models (MEMMs) [26], and conditional random fields (CRFs) [18,21], have been used for identifying semantic patterns in texts. These models are derived from either a joint probability function or a conditional probability function for a sequence of class assignments given a sequence of symbols These assumptions might not be the best assumptions for capturing such semantic patterns in texts . For example, one of the conditional independence assumptions is that the class identification for the current symbol depends only on the class identification of the previous symbol, not others.

Moreover, these graphical models lead to an optimization that dependently threads through the sequence of class assignments to optimize the joint or conditional probability of the class assignment given the symbol sequence. This optimization can be understood as an optimization of expected gain [11]. The implicit gain function employed by these models gains one if all class assignments are correct and zero if any assignment is wrong. No partial credit is given for some correct assignments. This criterion leads to difficulties where noise in data can cause the resulting optimal class assignments to hallucinate an incorrect yet seem-

* Corresponding author.
E-mail address: minhua.huang@tamucc.edu (M. Huang).

https://doi.org/10.1016/j.patrec.2018.03.008
0167-8655/© 2018 Elsevier B.V. All rights reserved.

ingly coherent class identification sequence. A noisy or perturbed symbol at any position in the input sequence can produce a wrong category path for a substantial length subsequence. In contrast an optimization using a gain function that gives credits for each correctly classified symbol, wrong but coherent classification subsequences of any substantial length tend not to happen.

Furthermore, for these graphical models, the maximum global probability value cannot be determined until the last symbol of the sequence has been reached and the computation must be done by a dynamic programming algorithm. In order to identify a sequence of n symbols among a set of m classes, these models need to have time complexity of $O(m^2n)$ and memory complexity of $O(mn)$.

By observing text data carefully, we have noticed that a word (or a symbol) in a sentence itself might carry lexical, semantic, or syntactic information. These pieces of information can be used to replace conditional dependences in existing methods. For example, in the case of NP chunking, identifying noun phrases in a sentence, information carried by the previous symbol of each symbol carries more class information about the current symbol than does the assigned class information associated with the previous symbol.

In contrast to these probabilistic graphical models, our model has fewer conditional independence assumptions. The true identification of the current symbol is based on information carried by the current symbol, the information between the current symbol and the preceding symbol, and information between the current symbol and the succeeding symbol. Moreover, understanding our optimization as an optimization of expected gain, our implicit gain function gives a partial credit for each correct class assignment. When we make a mistake on one symbol in a sequence, it will not effect other correct decisions that have been made or will be made for other symbols. Furthermore, our method does not need to employ dynamic programming. The time complexity is reduced to $O(mn)$ while the memory complexity is reduced to $O(n+m)$, Numerical comparisons are shown on Section 4.

We have applied the method to identify semantic patterns in documents [14–17]. The semantic patterns in documents are noun phrases, the meaning of a polysemous word , and semantic arguments of a verb in a sentence. In Section 5, we will formerly discuss the experiments. The results demonstrate that the method works well. For instance, the method achieves an average precision of 97.7% and an average recall of 98.8% for recognizing noun phrases on WSJ data from Penn Treebank; an average accuracy of 81.12% for recognizing the six sense word *line*; an average precision of 92.96% and an average recall of 94.94% for classifying semantic argument boundaries of a verb in a sentence on WSJ data from Penn Treebank and PropBank.

2. The method

2.1. Defining the task

Let $S = < s_1, \ldots, s_N >$ be a sequence of N symbols. Let C be a set of M classes, $C = \{C_1, \ldots, C_M\}$.[1] The task is to assign each symbol $s_n \in S$ a class $c_n^* \in C$, s.t. the sequence of classes $< c_1^*, \ldots, c_N^* >$ best describes $S = < s_1, \ldots, s_N >$ in the sense of

$$< c_1^*, c_2^*, \ldots, c_N^* > = \underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N | s_1, \ldots, s_N)$$

In order to find the optimal sequence $< c_1^*, c_2^*, \ldots, c_N^* >$, we need to compute the function of $p(c_1, c_2, \ldots, c_N | \; s_1, s_2, \ldots, s_N)$. Therefore, we build a decomposable graphical model [20].

[1] $C_m \in C$ will have a different meaning for each of the different tasks.

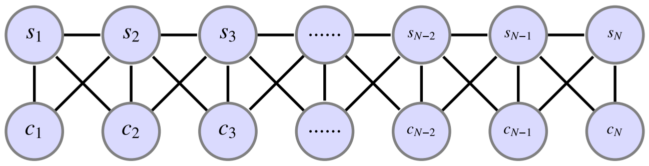


Fig. 1. The conditional independence graph defining our graphical model.

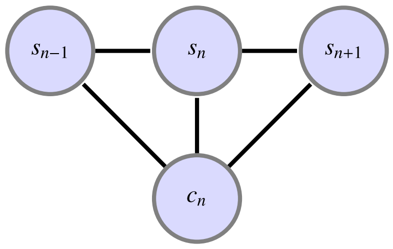


Fig. 2. The partial conditional independence graph for c_n.

2.2. The probabilistic graphical model

To find the assigned class sequence $c_1, \ldots, c_N$ for the input sequence $s_1, \ldots, s_N$ that maximizes the expected economic gain [11], we select a gain function that gains some value if a class assignment for a symbol in the input sequence is correct and zero otherwise. Based on Appendix A, we find an assigned class $c_n, n = 1, \ldots, N$ that maximizes the joint probability function $p(c_n, s_1, \ldots, s_N)$. Further, we assume that the current assigned class is only dependent on the current symbol, the preceding symbol, and the succeeding symbol. These lead to the graphical representation of our model in Fig. 1.

The graphical model in Fig. 1 leads to the mathematical representation in equation (1). See Appendix B and Appendix C.

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

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

To find a class sequence $< c_1^*, \ldots, c_N^* >$ for a symbol sequence $< s_1, \ldots, s_N >$, we only need to find c_n^* for s_n individually. Moreover, the denominator in (1) is a constant and it does not effect a decision for assigning c_i to s_i. Therefore,

$$< c_1^*, c_2^*, \ldots, c_N^* > = \prod_{n=1}^N \underset{c_n \in C}{argmax}\, p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n) \tag{2}$$

This simplifies for each n,

$$c_n^* = \underset{c \in C}{argmax}\, p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c) \tag{3}$$

2.4. Complexities

2.4.1. Time complexity

For each $p(s_{n-1}|s_{n+1}, s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$, we need to have three multiplications. To obtain the maximum probability value among M classes, we need to have $M-1$ comparisons. For a sequence of N symbols, we need

$$T_N = 3 * N * (M-1) = O(NM)$$

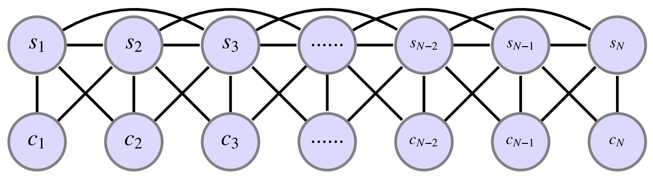


Fig. 3. The conditional independence graph defining the revised graphical model.

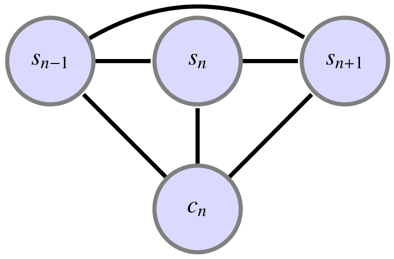


Fig. 4. The partial conditional independence graph is a complete graph of $\mathcal{K}_4$.

2.4.2. Memory complexity

For each symbol, we need to store M values of $p(s_{n-1}|s_{n+1}, s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$ to determine the maximal probability value. Hence, for a sequence of N symbols, we have

$$M_N = N + M = O(N + M)$$

3. A revised probabilistic graphical model

In a conditional independence graph, s_i and c_i, $i = 1, \ldots, N$, are nodes. Therefore, there are $2N$ nodes in total. If no conditional independence assumptions are made, there is a link between every pair of nodes. The degree of each node should be $2N - 1$. Compared to the conditional independence graph in Fig. 1, the conditional independence graph in Fig. 3 has less conditional independence assumptions.

$$p(c_1, \ldots, c_N \mid s_1, \ldots, s_N) = \frac{\prod_{n=1}^{N} p(s_{n-1}, s_n, s_{n+1}, c_n)}{\sum_{c_k \in C} \prod_{k=1}^{N} p(s_{k-1}, s_k, s_{k+1}, c_k)} \tag{4}$$

4. Comparisons

4.1. Graphical representations and dependence sequences

All existing graphical models are derived under some conditional independence assumptions. Fig. 5 presents an HMM [26,28], an MEMM [26], a CRF [21,30], a Naive Bayes, and our models (For the simplicity, here only shows links for c_i). While HMM and MEMM are directed graphical models, Naive Bayes, CRF, and our model are undirected graphical models. Compared with these graphical models, we note that for each c_i, the degree of our model is three while others are at most two. This indicates that our model has fewer 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 Naive Bayes is built under the assumption of given the current symbol, the current class is independent of other classes and symbols. 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.

Obviously, c_i can be better predicated by our model than the Naive Bayes because our model has more information (more links) for predicating c_i than the Naive Bayes. Moreover, in HMM, MEMM, and CRF, the sequence dependency is a dependency of class to neighboring class. In our model, one dependency is between class and neighboring symbols and another dependency is a dependency of symbols to neighboring symbols. We believe that c_i can be better predicated from s_{i-1} and s_{i+1} rather than c_{i-1} when a symbol contains several types of information, such as lexical information and syntactical information. For example, in the case of NP chunking, Part-of-speech (POS) tag information carried on the previous symbol of each symbol is more useful than the class information assigned to the previous symbol.

4.2. Complexities

HMMs or CRFs employ dynamic programming to obtain a sequence of optimal 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. We compute ratios of time complexity and memory complexity of our model to *HMMs* and *CRFs* to see the differences.

The ratio of time complexity is $\frac{NM}{M^2N} = \frac{1}{M}$ The ratio of memory complexity is $\frac{M+N}{M*N} = \frac{1}{N} + \frac{1}{M}$ If we need to recognize a sequence of N symbols with M categories, our model only takes $\frac{1}{M}$ time and memory compared to HMMs or CRFs. For example, if the cardinality of C is $(M = 6)$, for a sequence of fifty symbols $(N = 50)$, our method only needs to have $\frac{1}{6}$ time and $\frac{14}{75}$ memory of a HMM or a CRF to recognize this sequence.

5. Application – recognizing text patterns

The model discussed in Section 2.1 is applied to classify three types of text patterns. These text patterns are the sense of a polysemous word, noun phrases of a sentence and the semantic arguments of a verb. For identifying the word sense of a polysemous word, in contrast with the methods in papers [7,12,22,23,35], in our method, the polysemous word is represented by a sequence of context symbols, each symbol is an 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, for identifying NP, compared with the methods proposed by papers [1,5,28,29,34]. We follow Ramshaw's idea [29] of designing three categories for a word in a sentence to determine whether the word is inside an NP chunk, outside an NP chunk, or should start a new NP chunk. However, most methods for this task use HMMs [26,28], MEMMs [26], and CRFs [21,30].

For identifying semantic arguments of a verb from a sentence, our method is different from the methods proposed by papers [2,6,8–10]. Our method is developed based on the idea that if a sentence is represented by a rooted tree, the root is the start of a sentence and leaves are words in the sentence. A semantic argument of a verb in the sentence will be associated with a rooted subtree. Hence, all semantic arguments of a verb in the sentence will be represented by a set of rooted subtrees. For each verb node v, there exists a path, from which, all roots of the

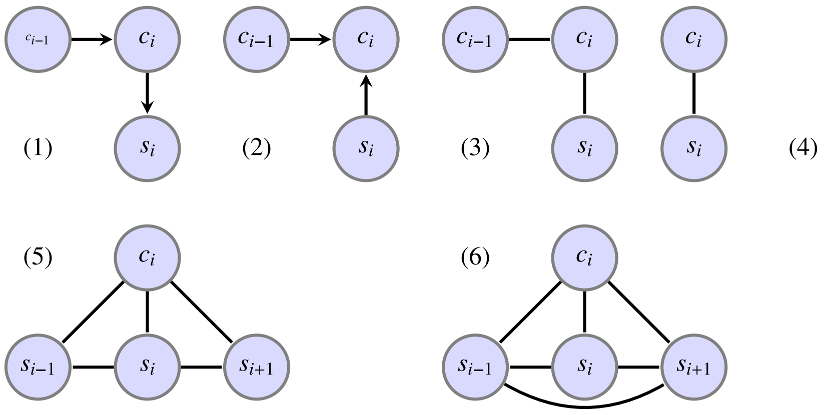


Fig. 5. This shows the conditional independence graphs for different models: (1): a HMM model, (2): a MEMM model, (3): a CRF model, (4): a Naive Bayes model, (5) and (6): the models presented by this paper.

subtrees will be extracted. Obviously, the unique feature, which is a path, represents all the semantic arguments of the verb.

5.1. Identifying the word sense of a polysemous word

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

5.1.1. Defining 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, $T_t = (t_1, \ldots, t_k, \ldots, t_K)$, $t_k \in S$, and $K \leq N$.

5.1.2. Identifying the sense of a word

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

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

5.2. Identifying NP chunks in a sentence

Let $S =< s_1, \ldots, s_i, \ldots, s_N >$ be a sequence of symbols associated with a sentence. 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.

5.2.1. 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 $\mathcal{B}' \subseteq \mathcal{B}$

5.2.2. Identifying noun phrases

- 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 $\mathcal{B}$.

5.3. Identifying semantic arguments of a verb

Let $T = (V, E, r)$ be a tree associated with a sentence, where V is a set of vertices, E is a set of edges, $E \subseteq V \times V$. Let $r \in V$ be the root of T. Each vertex $v \in V$ is associated with a label defined by Weischedel et al. [33]. Let π be a set of special labels related verbs in a sentence. Let $V' \subset V$, each $v' \in V'$ has a label in π. 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.

- Form a path of $\mathcal{P}(v)$, $v \in V'$
  - For $v \in V'$, and v is not a node in $\mathcal{P}(v')$, $\mathcal{P}(v')$ is a path that has been already formed previously, $v' \in V'$, find

    $$< v_1^*, \ldots, v_K^* >= \underset{v_1,\ldots,v_K}{argmax}\, p(c_1, \ldots, c_K, v_1, \ldots, v_K)$$

    Where, $c_k \in C$, $v_k \in V$, $\{v_{k-1} v_k\} \in E$.
- Form a set of roots $R(v) = \{r_i | i = 1 \ldots M\}$ from $\mathcal{P}(v)$, where $r_i \leq v_k$.
  - For all siblings of v_k, find x, s.t. $x \notin V'$ and $x \notin \{v_k | k = 1, \ldots, K\}$, then $R(v) \leftarrow R(v) \cup \{z\}$
  - For all children of v_k, find y, s.t. $y \notin V' \; and \; y \notin \{v_k | k = 1, \ldots, K\}$, then $R(v) \leftarrow R(v) \cup \{y\}$
- Find a rooted forest $F(v) = \{T_i | i \in \{1, \ldots, I\}\}$,
  - Each T_i is induced from the root r_i by all its codependents.
  - For each $T_i \in F(v)$, the leaves $\{l_i^1, \ldots, l_i^K\}$ correspond to one of the semantic arguments of v.

5.3.1. An example

- The sentence, *He sat on the bank of the river and watched the currents.*, represented by the rooted tree T is shown in Fig. 6.
- A path for verb *sit* is in Fig. 7.
- A labeled rooted forest $F(sit) = \{T_1, T_2\}$ is in Figs. 8 and 9.
- The semantic arguments of verb *sit* are:
  - *He*
  - *on the bank of the river*

6. Experiments

6.1. Data sets and evaluation metric

The data sets that we used for evaluating our methods were the WSJ data from the Penn TreeBank and the PropBank [27,33], data developed by Leacock et al. [23] and Bruce and Wiebe [3], and data of CoNLL-2000 Shared Task [31]. Our reasons for using these data sets were that they have been studied by a number of

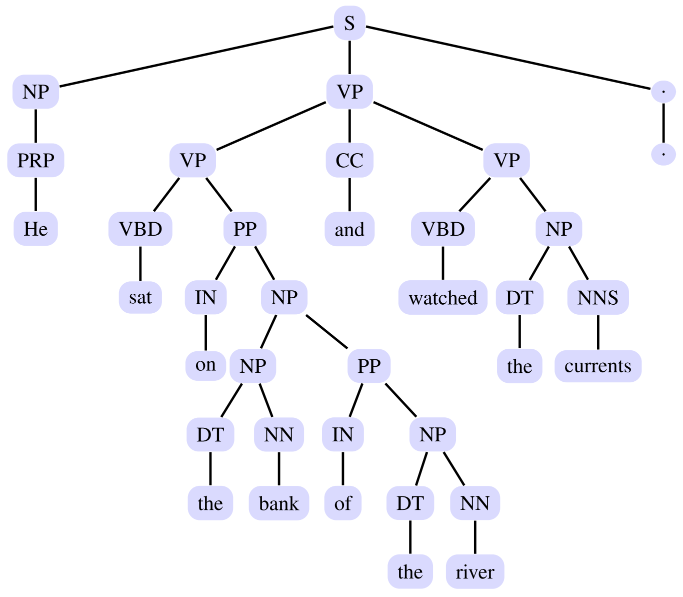


Fig. 6. The parse tree of the sentence: *He sat on the bank of the river and watched the currents.*

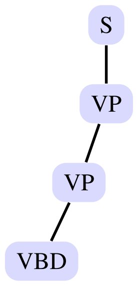


Fig. 7. The path for the verb *sit*.

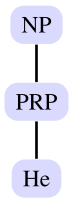


Fig. 8. The rooted tree T_1.

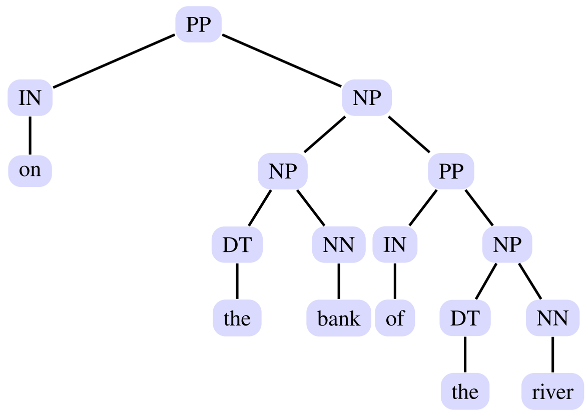


Fig. 9. The rooted tree T_2.

other researchers and many results have been published over the years. The evaluation metrics designed for classifying semantic arguments of a verb and NP chunks were *precision* , *recall*, *F-measure* (F_1). and for classifying the sense of a polysemous word was *accuracy*. The reason for selecting different evaluation methods was based on the design of classes described in Sections 5.1–5.3.[2] One of the classes needed no evaluation in the tasks described in Section 5.3 and 5.2 while all the classes needed to be evaluated in the task described in Section 5.1. Moreover, we have used a 10-fold cross validation technique for obtaining our result for all experiments.

6.2. Data preprocessing

To prepare data for recognizing the sense of a polysemous word and NP chunks in a sentence, capital letter words were converted into low case words. Punctuation marks were deleted. Digits were coerced into artificial words. Moreover, each word in a sentence was tagged with a part-of-speech tag. Furthermore, for the first task, for each given target word, the left *N* open class words and right *N* open class words were extracted, *N* was in the range of two to ten. To prepare data for recognizing the semantic arguments of a verb in a sentence, for each sentence, the corresponding parse tree [33] was obtained. Then the tree was transformed into an adjacency matrix and a corresponding label table.

6.3. Results and discussions

Table 1 shows the results of our method for identifying the sense of a polysemous word (the first text pattern), NP chunks (the second text pattern), and the semantic arguments of a verb (the third text pattern). Details will be discussed as the following sections.

6.3.1. Recognizing the sense of given target words

Experiments were conducted for identifying the sense of a polysemous word on data sets *line, interest, serve*, and *hard*. The senses' descriptions and instances' distributions could be found in [3,23]. 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, *interest* had 3 senses (other 3 senses were omitted due to lack of instances). 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 row one to row five in Table 1. A base line at the last column was obtained by assigning the most frequent sense to a given target word in test samples.

In this experiment, we found that misclassified instances were primarily generated by the ambiguity of context words. For example in Table 1, 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 averaging 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 the accuracies generated from each combination for the word *line*, we found that some combination, for example, (Sense#1, Sense#2, Sense#4) had the highest average accuracy: 91.7% while some combination, for example, (Sense#1, Sense#3, Sense#5) had lowest average accuracy: 77.1%. The difference was almost 20%. By carefully checking these misclassified instances, We learned that

[2] There was no class that represented none of these classes for the previous two patterns while every class was a distinct sense for the second pattern.

Table 1
Output results of our method for identifying three different text patterns.

Text Pattern	Data	Symbol type	Precision %	Recall %	F-measure Accuracy%	Base line %
1	line [3]	Lexicon+POS			81.16	16.67
1	line [4]	Lexicon+POS			85.25	33.33
1	interest	Lexicon+POS			92.10	33.33
1	serve	Lexicon+POS			79.80	25.00
1	hard	Lexicon+POS			82.88	33.33
2	CoNLL	Lexicon + POS	95.15	96.05	95.59	
		POS	92.27	93.76	92.76	
		Lexicon	86.27	93.35	89.75	
2	WSJ	Lexicon+POS	97.73	98.65	98.18	
3	WSJ	Labeled node [5]	92.34	94.17	93.25	

4: The *line* has six senses. 5: The *line* has three senses. 6: Defined by [27] and [33].

Table 2
Comparisons on recognizing word sense on **line** data.

Method	Bayesian	Our Method	LSA	CV	NN
Accuracy%	71	81	75	72	76

if two senses are similar to each other, there were more chances that their contexts consisted of the same words. As a consequence, the misclassification rate increases.

We average the results from the ambiguous nouns, the ambiguous adjective, and the ambiguous verb in Table 1, our model achieves an average of accuracy 83.5184%. In Table 2, we show the comparisons between our method on the six sense word *line* with methods LSA by Levin et al. [24], CV and NN by Leacock et al. [23]. The result achieved by our method was very encouraging and surpassed the results published by these methods. Moreover, by observing the outputs of two polysemous nouns *line* and *interest*, we found that as the number of senses of a polysemous noun increases, the accuracy decreases. This suggests that nouns with a larger number of senses are more difficult to recognize than nouns with small number of senses by our method. Furthermore, by observing accuracies in column six, we noticed that nouns were relatively easier to identify than adjectives or verbs. Moreover, we noticed that accuracies generated by our method on adjectives had a larger variance than that on nouns or verbs.

6.3.2. *Recognizing the sense of word* line *by the revised model*

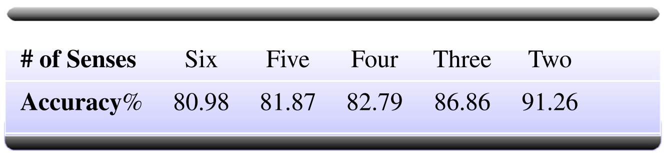
We test the revised probabilistic graphical model described in Section 3 on the collection of sets of senses with the size of two, three, four, five, and six on the word *line*. To obtain the value of $\prod_{n=1}^{N} p(s_{n-1}|s_{n+1}, s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$, especially the value of $p(s_{n-1}|s_{n+1}, s_n, c_n)$ on the data set, we compute $\sum_{n=1}^{N}(logp(s_{n-1}|s_{n+1}, s_n, c_n) + logp(s_{n+1}|s_n, c_n) + logp(s_n|c_n) + logp(c_n))$ instead. For each case, we compute all the combinations of $\binom{6}{n,}$ where $n = 2, 3, 4, 5, 6$. We average the results. The average accuracy for two senses is 91.26%, the average accuracy for three senses is 86.85%, the average accuracy for four senses is 82.76%, the average accuracy for five senses is 81.87%, and the average accuracy for six senses is 80.98%. Compared with Tables 1 and 3, we notice that in the case of the word *line* has three senses, the Revised model achieved the better result. However, in the case of the word *line* has six senses, the model discussed in Section 2.2 achieved the better result. For this model, We need to do further study.

6.3.3. *Recognizing NP chunks in a sentence*

Experiments were conducted for identifying NP chunks on CoNLL-2000 data and WSJ data from Penn Treebank. Three types of symbols designed for identifying NP chunks 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 are shown in the 6th row of Table 1. Observing the column six, we notice that if we only use the lexical information, the method had the lowest performance 89.75%. The method's performance improved 3% if we use only POS tags. The method achieved the best performance of 95.59% if both lexicon and POS tags were included.

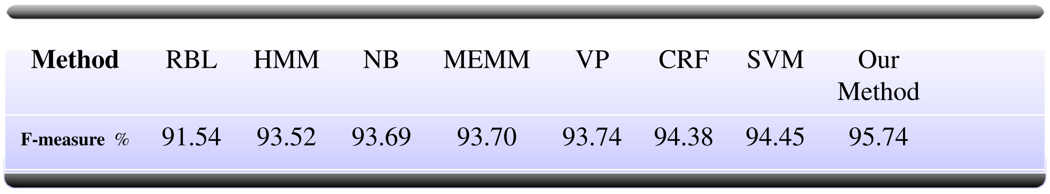
On the second experiment on the WSJ data from Penn Treebank, shown at 7th row of Table 1, used only one type of symbol: the lexicon and the POS tag of a word. The main reason for using this data set was that we wanted to see whether the performance of our model could be improved when it was built on 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 row seven of Table 1. In this experiment, the standard deviations of precision, recall, and *F*-measure were 0.19, 0.14, and 0.08. Comparing the results on these two data sets, we notice that the average precision improved about 2.7% from

Table 3
Recognizing word sense on **line** data on revised model.



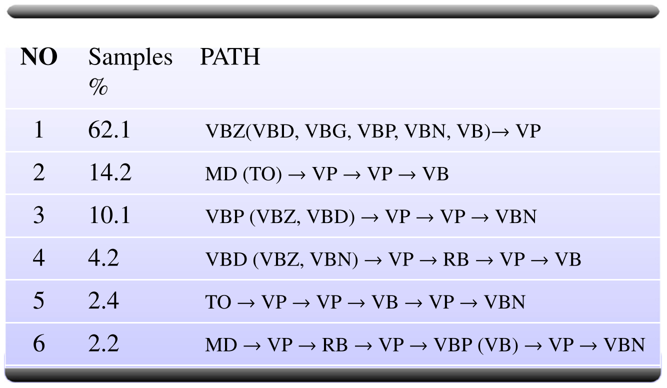
# of Senses	Six	Five	Four	Three	Two
Accuracy%	80.98	81.87	82.79	86.86	91.26

Table 4
Comparisons on recognizing NP chunks on CoNLL-2000 data.



Method	RBL	HMM	NB	MEMM	VP	CRF	SVM	Our Method
F-measure %	91.54	93.52	93.69	93.70	93.74	94.38	94.45	95.74

Table 5
Six typts of paths.



NO	Samples %	PATH
1	62.1	VBZ(VBD, VBG, VBP, VBN, VB)→ VP
2	14.2	MD (TO) → VP → VP → VB
3	10.1	VBP (VBZ, VBD) → VP → VP → VBN
4	4.2	VBD (VBZ, VBN) → VP → RB → VP → VB
5	2.4	TO → VP → VP → VB → VP → VBN
6	2.2	MD → VP → RB → VP → VBP (VB) → VP → VBN

95.15% to 97.73% . The average recall improved about 2.8% from 96.05% to 98.65%. The average *F*-measure improved about 2.7% from 95.59% to 98.2% as the training sets expanded to seven times larger. This suggests that the larger the training set, the better the performance that was obtained by our method.

Table 4 showed the best performances of the related methods on the CoNLL-2000 shared task data. Compared with the method RBL by Veenstra and den Bosch [32], the method HMM by Molina et al. [28], the method NB (We implemented the Naive Bayse based on the equation $p(s_1, \ldots, s_N|c_1, \ldots, c_N) = \prod_{n=1}^{N} p(s_n|c_n)$.), the method MEMM by Sha and Fereira [30], the method VP [4], the method CRF [30], the method SVM [34], we see that the role based learning achieved the worst *F*-measure performance and our method achieved the best *F*-measure performance.

6.3.4. Recognizing semantic arguments of a verb in a sentence

Experiments were conducted for identifying semantic arguments of a verb in a sentence on the data set, the section 00 of WSJ from Penn Treebank and PropBank [33]. There were 223 sentences in files 20, 37, 49, and 89. Associated with each of these sentences, 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 Table 5. We noticed that 86% paths fell into the first three patterns. 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, was formed. The test results were shown in Table 1. On the average, each time $\frac{1}{10}$ of the semantic arguments were classified, about 93% of semantic arguments was correctly identified and 7% of semantic arguments was mistakenly identified. By checking these classified instances, we found that our system 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 system was less effective. The reason was that these phrases were parts of leaves of a tree induced from a root determined by our system. This suggested that in order to exclude phrases from a semantic argument, we need to develop a method so that a set of subroots can 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.

7. Conclusions

A new generic probabilistic graphical model has been discussed throughout this paper. It has an unique graphical representation which is different from other existing graphic models such as HMMs, CRFs, and MEMMs. It does not need to employ dynamic programming for obtaining a sequence of optimal class assignments for a sequence of symbols. As a consequence, it requires less computation time and less memory than other competing techniques. Moreover, because a sequence of optimal class assignments for a sequence of symbols is determined by finding the optimal class assignment for each symbol independently, the misclassification for the sequence of class assignments is reduced. This model is applied to recognize three types of text patterns. These patterns are noun phases, the sense of a given polysemous word, and semantic arguments of a verb. The results achieved by our methods have demonstrated the effectiveness of our graphical model.

Appendix A

Economic gain function. Let $s =< s_1, \dots, s_N >$ be a sequence of N symbols. Let C be a set of M classes, $C = \{C_1, \dots, C_M\}$. Let $c =< c_1, \dots, c_N >$ be a sequence of assigned classes. Let $c^T =< c_1^T, \dots, c_N^T >$ be a sequence of true class classes. Let e be the economic gain function $e : C^N \times C^N \to \mathcal{R}^+$. It is defined by:

$$e(c_1^T, \dots, c_N^T, c_1, \dots, c_N) = \sum_{n=1}^{N} e(c_n, c_n^T) \tag{5}$$

where

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

To maximize the expected gain under equation (5):

$$\begin{aligned} &E[e(c_1, \dots, c_N)] \\ &= \sum_{c_1^T,\dots,c_N^T} e(c_1^T, \dots, c_N^T, c_1, \dots, c_N) p(c_1^T, \dots, c_N^T, s_1, \dots, s_N) \\ &= \sum_{c_1^T,\dots,c_N^T} \sum_{n=1}^{N} e(c_n^T, c_n) p(c_1^T, \dots, c_N^T, s_1, \dots, s_N) \\ &= \sum_{n=1}^{N} \sum_{c_1^T,\dots,c_N^T} e(c_n^T, c_n) p(c_1^T, \dots, c_N^T, s_1, \dots, s_N) \\ &= \sum_{n=1}^{N} \sum_{c_n^T} e(c_n^T, c_n) \sum_{c_1^T,\dots,c_{n-1}^T,c_{n+1}^T,\dots,c_N^T} p(c_1^T, \dots, c_N^T, s_1, \dots, s_N) \\ &= \sum_{n=1}^{N} \sum_{c_n^T} e(c_n^T, c_n) p(c_n^T, s_1, \dots, s_N) \end{aligned}$$

When the gain matrix is diagonal and positive, then:

$$E[e(c_1, \dots, c_N)] = \sum_{n=1}^{N} e(c_n, c_n) p(c_n, s_1, \dots, s_N)$$

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

$$E[e(c_1, \dots, c_N)] = \sum_{n=1}^{N} p(c_n, s_1, \dots, s_N)$$

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

$$max(E[e(c_1, \dots, c_N)]) = \sum_{n=1}^{N} \max_{c_n \in C} p(c_n, s_1, \dots, s_N)$$

Appendix B

Statistical graphical models. A graph contains nodes and links. In a probabilistic graphical model, each node represents a random variable (or a group of random variables) and links carry probabilistic relationships between these variables. We say that node A and node B are independent if there is no link between them. We say that node A and node B are conditional independent on the node C if A reaches B (or B reaches A) through C. A probability distribution can be represented as a directed graphical model or an undirected graphical model. The following is an example. The probability distribution $p(A, B, C) = p(C)p(B|C)p(A|BC)$ represented by a directed graphical model is depicted in the left hand side of Fig. 10. An undirected graphical model is depicted in the right hand side of Fig. 10.

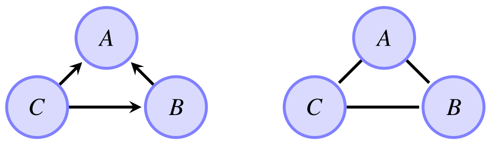


Fig. 10. $p(A, B, C) = p(C)p(B|C)p(A|BC)$.

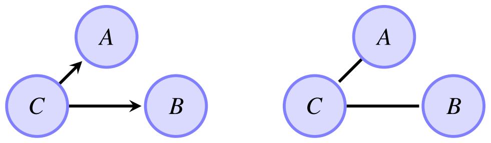


Fig. 11. $p(A, B, C) = p(C)p(A|C)p(B|C)$.

If we assume that A is independent of B given C, then we have $p(A, B, C) = p(C)p(A|B)p(B|C)$. This model can be described by Fig. 11.

Note, in an undirected graphical model, the probability distribution of all random variables equals a product of probabilities of cliques from the clique set: $\{\Lambda_1, \dots, \Lambda_N\}$ divided by a product of probabilities of separators from the separator set: $\{\Gamma_1, \dots, \Gamma_M\}$. Let $G = (V, E)$, where V is a set of nodes and E is a set of edges, s.t. $E \subseteq V \times V$. A clique is a maximal complete set of nodes, s.t. $\Lambda \subseteq V$ is a clique if and only if $\lambda_1,\ \lambda_2 \in \Lambda$, $\lambda_1 \neq \lambda_2$, imply $\{\lambda_1,\ \lambda_2\} \in E$ and there is no set that properly contains Λ with this property. $\Gamma = \{\Gamma_1, \dots, \Gamma_M\}$ is a set of separators, where $\Gamma_k = \Lambda_k \cap (\Lambda_1 \cup \dots \cup \Lambda_{k-1})$. For example, in the right hand side of Fig. 11, the cliques are $\{B,\ C\}$ and $\{A,\ C\}$ and the separator is $\{B, C\} \cap \{A, C\} = \{C\}$. Therefore:

$$\begin{aligned} p(A, B, C) &= \frac{p(A, C)p(B, C)}{p(C)} = p(A, C)p(B|C) \\ &= p(C)p(A|C)p(B|C) \end{aligned}$$

Appendix C

Computing $p(c_1, \dots, c_N | s_1, \dots, s_N)$

From Fig. 1, according to appendix A, we have:

$$\begin{aligned} &p(c_1, \dots, c_N, s_1, \dots, s_N) \\ &= \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n) p(s_n, s_{n+1}, c_{n+1})}{\prod_{m=1}^{N-1} p(s_m, s_{m+1}) \prod_{m=2}^{N-1} p(s_m, c_m)} \\ &= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \\ &\quad \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1}) \\ &= \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=2}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \\ &\quad \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1}) \end{aligned}$$

$$= \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n) \times \prod_{m=2}^{N} p(s_{m-1}, s_m, c_m)$$

Setting $n = m - 1$ in the last product:

$$p(c_1, \ldots, c_N, s_1, \ldots, s_N) = \frac{p(s_1, s_2, c_1)p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n) \times \prod_{m=2}^{N-1} p(s_{m-1}, s_m, c_m)$$

$$= \frac{p(s_1, s_2, c_1)p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n)p(s_{n-1}, s_n, c_n)$$

$$= \frac{p(s_1, s_2, c_1)p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n)p(s_{n-1}|s_n, c_n)p(s_n|c_n)p(c_n)$$

$$= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times p(s_2|s_1, c_1)p(s_1|c_1)p(c_1) \times p(s_{N-1}|s_N, c_N)p(s_N|c_N)p(c_N) \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n)p(s_{n-1}|s_n, c_n)p(s_n|c_n)p(c_n)$$

Define $p(s_0|s_1, c_1) = 1$ and $p(s_{N+1}|s_N, c_N) = 1$, then:

$$p(c_1, \ldots, c_N, s_1, \ldots, s_N) = \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times p(s_2|s_1, c_1)p(s_0|s_1, c_1)p(s_1|c_1)p(c_1) \times p(s_{N+1}|s_N, c_N)p(s_{N-1}|s_N, c_N)p(s_N|c_N)p(c_N) \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n)p(s_{n-1}|s_n, c_n)p(s_n|c_n)p(c_n)$$

$$= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$$

Define $M_{s_1,\ldots,s_N} = \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})}$, then:

$$p(c_1, \ldots, c_N, s_1, \ldots, s_N) = M_{s_1,\ldots,s_N} \times \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$$

Therefore the conditional probability:

$$p(c_1, \ldots, c_N|s_1, \ldots, s_N) = \frac{p(c_1, \ldots, c_N, s_1, \ldots, s_N)}{\sum_{c'_1,\ldots,c'_N \in C} p(c'_1, \ldots, c'_N, s_1, \ldots, s_N)} = \frac{\prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}{\sum_{c_k \in C} \prod_{k=1}^{N} p(s_{k-1}|s_k, c_k)p(s_{k+1}|s_k, c_k)p(s_k|c_k)p(c_k)}$$

Appendix D

Knowledge of a rooted tree. A rooted tree T is a 3-tuple (V, E, r), where V is a finite set of vertices, $E \subseteq V \times V$ is a finite set of edges, and $r \in V$ is the root that all edges of T are directed away from it. The tree-order is the partial ordering on V for any v, $u \in V$, $u \le v$ if and only if the unique path from the root r to v passes through u. In T, the root r is a unique minimal vertex and we say it has level 0. An edge in E is an ordered pair $(x, y) \in (V \times V)$ s.t. $x < y$ and there exists no $z \in V$ with $x < z < y$. In this case, x is a parent of y and y is a child of x. If two nodes[3] x, y have the same parent z, x and y are called siblings. Any node y on the unique path from r to x is called an ancestor of x. In this case, x is a descendant of y. The sub-tree rooted at node x is the tree induced by the descendants of x. A node with no children is an external node or a leaf. A node that is not a leaf node is an internal node. The largest depth of a node in T is the *height* of T. A rooted forest is a set of rooted trees, s.t. $F = \{T_i | i = 1 \ldots N\}$ where T_i is a rooted tree.

References

[1] S. Abney, S.P. Abney, Parsing by chunks, in: Principle-Based Parsing, Kluwer Academic Publishers, 1991, pp. 257–278.
[2] U. Baldewein, K. Erk, S. Pad, D. Prescher, Semantic role labeling with chunk sequences, in: Proceedings of CoNLL-2004 Shared Task, 2004.
[3] R. Bruce, J. Wiebe, Word-sense disambiguation using decomposable models, in: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994, pp. 139–146.
[4] X. Carreras, L. Mrquez, Phrase recognition by filtering and ranking with perceptrons, in: the International Conference on Recent Advances on Natural Language Processing, 2003.
[5] K.W. Church, A stochastic parts program and noun phrase parser for unrestricted text, in: Proceedings of the Second Conference on Applied Natural Language Processing, 1988, pp. 136–143.
[6] T. Cohn, P. Blunsom, Semantic role labelling with tree conditional random fields., in: Proceedings of CoNLL-2005 Shared Task, 2005.
[7] W. Gale, K. Church, D. Yarowsky, A method for disambiguating word senses in a large corpus, in: Computers and the Humanities, 1992, pp. 415–439.
[8] D. Gildea, D. Jurafsky, Automatic labelling of semantic roles, Comput. Linguist. (2002) 245–288.
[9] K. Hacioglu, A semantic chunking model based on tagging, in: Proceedings of HLT/NACCL-2004, 2004.
[10] K. Hacioglu, Semantic role labeling using dependency trees, in: Proceedings of Coling 2004, COLING, Geneva, Switzerland, 2004, pp. 1273–1276.
[11] R.M. Haralick, L.G. Shapiro, Computer and Robot Vision, vol. 1, Addison-Wesley, 1991.
[12] M.A. Hearst, 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, pp. 1–22.
[13] S. Hochreiter, J. Schmidhuber, Long-short term memory, Neural Comput. 9 (1997) 1735–1780.
[14] M. Huang, R.M. Haralick, Recognizing Patterns in Texts, River, 2010.
[15] M. Huang, R.M. Haralick, Discovering text patterns by a new graphic model, in: 7th International Conference on Machine Learning and Data Mining, 2011, pp. 428–442.
[16] M. Huang, R.M. Haralick, Discovering text patterns by a new graphic model, J. Trans. Mach. Learn. Data Mining 4 (2011) 55–74.
[17] M. Huang, R.M. Haralick, Developing an algorithm for mining semantics in texts, in: 13th International Conference on Intelligent Text Processing and Computational Linguistics, 2012, pp. 247–260.
[18] A.N. Jagannatha, H. Yu, Structured prediction models for rnn based sequence labeling in clinical text, in: EMNLP, 2016, pp. 856–865.
[19] I.B.J. Jingle, J.J.A. Celin, Markov model for discovering knowledge in text documents, J. Theor. Appl. Inf. Technol. 70 (2014) 459–463.
[20] D. Koller, N. Friedman, Probabilistic Graphical Models Principles and Techniques, The MIT Press Cambridge, 2009.
[21] J. Lafferty, A. MaCallum, F. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in: Proceedings of 18th International Conf. on Machine Learning, 2001, pp. 282–289.
[22] C. Leacock, G.A. Miller, M. Chodorow, Using corpus statistics and wordnet relations for sense identification, Computat. Linguist. 24 (1998) 147–165.
[23] C. Leacock, G. Towell, E. Voorhees, Corpus based statistical sense resolution, in: Proceedings of the workshop on Human Language Technology, 1993, pp. 260–265.
[24] E. Levin, M. Sharifi, J. Ball, Evaluation of utility of lsa for word sense discrimination, in: Preceedings of HLT-NAACL, 2006, pp. 77–80.
[25] H. Li, Deep learning for natural language processing: advantages and challenges, Natl. Sci. Rev. (2017).
[26] A. MaCallum, D. Freitag, F. Pereira, Maximum entropy Markov models for information extraction and segmentation, in: Proceedings of 17th International Conference on Machine Learning, 2000, pp. 591–598.

[3] In a rooted tree, a vertex can be also called a node.

[27] M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of english: the penn treebank, Computat. Linguist. 19 (2) (1994) 313–330.
[28] A. Molina, F. Pla, J. Hammerton, M. Osborne, S. Armstrong, W. Daelemans, Shallow parsing using specialized HMMs, J. Mach. Learn. Res. 2 (2002) 595–613.
[29] L.A. Ramshaw, M.P. Marcus, Text chunking using transformation-based learning, in: Proceedings of the Third Workshop on Very Large Corpora, 1995, pp. 82–94.
[30] F. Sha, F. Fereira, Shallow parsing with conditional random fields, in: Proceedings of HLT-NAACL, 2003, pp. 213–220.
[31] E.F. Tjong, K. Sang, Introduction to the conll-2000 shared task: chunking, in: Proceedings of CoNLL-2000, 2000, pp. 127–132.
[32] J. Veenstra, A.V. den Bosch, Single-classifier memory-based phrase chunking, in: Preceedings of CoNLL-2000 and LLL-2000, 2000, pp. 157–159.
[33] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, 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.
[34] Wu-Chieh, Wu, Y.-S. Lee, J.-C. Yang, Robust and efficient multiclass svm models for phrase pattern recognition, Pattern Recognit. 41 (2008) 2874–2889.
[35] D. Yarowsky, Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French, in: Proceedings of the 32nd Annual Meeting, 1994.
[36] W. Yin, K. Kann, M. Yu, H. Schutze, Comparative study of CNN and RNN for natural language processing, CoRR (2017). abs/1702.01923