Developing Probabilistic Models for Identifying Semantic Patterns in Texts

Minhua Huang
Computer Science Department
The Graduate School and University Center
The City University of New York
New York, NY 10016
mhuang@gc.cuny.edu

Robert M. Haralick
Computer Science Department
The Graduate School and University Center
The City University of New York
New York, NY 10016
haralick@aim.com

Abstract

We present a probabilistic graphical model that finds a sequence of optimal categories for a sequence of input symbols. Based on this mode, three algorithms are developed for identifying semantic patterns in texts. They are the algorithm for extracting semantic arguments of a verb, the algorithm for classifying the sense of an ambiguous word, and the algorithm for identifying noun phrases from a sentence. Experiments conducted on standard data sets show good results. For example, our method achieves an average precision of 92.96% *and an average recall of* 94.94% *for extracting semantic argument boundaries of verbs on WSJ data from Penn Treebank and PropBank; an average accuracy of* 81.12% *for recognizing the six sense word 'line'; and an average precision of* 97.7% *and an average recall of* 98.8% *for recognizing noun phrases on WSJ data from Penn Treebank.*

1 Introduction

Graphical models that lead to an optimization must dependently thread through the sequence of class assignments to optimize the joint probability of the class assignment given the measurements, essentially use an implicit gain function that 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. This criterion leads to difficulties where noise in the text data extraction can cause the resulting optimal class assignments to hallucinate an incorrect yet seemingly coherent result. In effect what happens here is that a noisy or perturbed symbol at any position in the input sequence can produce a wrong category path for the whole sequence.

This probabilistic graphical model presented in the paper differs from existed graphical models: CRFs [5], HMMs [9], MEMMs [8]. It gives partial credit and yet takes class dependencies into account. The model is derived from the probability function of a sequence of categories given a sequence of symbols by using the information carried on each current symbol, the association between the current symbol and the preceding symbol, and the association between the current symbol and the succeeding symbol. Several benefits result. First, we do not need to compute all category paths and store them in order to determine the optimal category path once the last symbol of a sequence has been reached. As a consequence, for recognizing a new symbol sequence, the time complexity is reduced from $O(M^2N)$ to $O(MN)$ while the memory complexity is reduced from $O(MN)$ to $O(M)$ comparing with dynamic programming. Furthermore, when we made 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. Therefore, the ratio of misclassification for the whole sequence of categories can be reduced. Indeed, this is the behavior we have observed using this kind of model for three different types of text pattern recognition.

2 Developing the Model

Let $S =< s_1, \ldots, s_N >$ be a sequence of N symbols. Let C be a set of M categories, $C = \{C_1, \ldots, C_M\}$. We need to find $< c_1^*, \ldots, c_N^* >$, $c_n^* \in C$ that best describes $S =< s_1, \ldots, s_N >$.

$$< c_1^*, c_2^*, ..., c_N^* >= \underset{c_1, c_2, \ldots, c_N}{argmax}\ p(c_1, c_2, \ldots, c_N | s_1, s_2, \ldots, s_N)$$

Our graphical model leads to the following representation for the probability [3] [2] [4].

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

Because of the denominator is a constant for all $c_k \in C$, $k = 1...N$, therefore,

$$\underset{c_1, .., c_N}{argmax}\, p(c_1, c_2, \ldots, c_N | s_1, s_2, \ldots, s_N)$$

978-0-7695-4492-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ICSC.2011.35

$$= \underset{c_1,..,c_N}{argmax} \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)$$
$$= \prod_{n=1}^{N} \underset{c_n \in C}{argmax}\, p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n) \quad (2)$$

2.1 Complexities

Time Complexity. By equation (1), to determine the c_n for a s_n, we need to have four multiplications and $M-1$ comparisons, where M is the cardinality of C. In the case of a sequence of N symbols, we will have the time complexity:

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

Memory Complexity. Because the global maximum probability is determined by each local maximal probability, for a sequence of N symbols, we only need to store the information of the current symbol. Therefore,

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

3 Three Algorithms

Based on the model, we develop three algorithm for three different tasks. Algorithm 1 is for extracting semantic arguments of a verb; algorithm 2 is for classifying the sense of an ambiguous word; and algorithm 3 is for identifying noun phrases from a sentence.

3.1 Algorithm 1

Let $T = (V, E, r, A, L)$ be a labeled rooted tree associated with a sentence, where V is a set of nodes, E is a set of edges, $E \subseteq V \times V$, r is the root, L is a function, s.t. $L : V \rightarrow A$, A is defined by [11]. Let π be a set of labels related verbs, s.t. $\pi \subseteq A$. Let $C = \{C_1, C_2\}$ be a set of class categories, where C_1 represents there is a path from the current node to an adjacent node; C_2 represents there is not a path from the current node to an adjacent node. The algorithm can be stated as follows:

- Apply equation (1) to find a path for each verb node x, s.t. $\mathcal{P}(x) = \tau_1 \rightarrow \tau_2 \ldots \rightarrow \tau_K$, where $x \in V$, $L(x) \in \pi$, and x is not a node in $\mathcal{P}'(y)$, $\mathcal{P}'(y)$ is a path that has already been formed. Each $\tau_k \in V$, $k = 1, ..., K$
- Find a set of roots $R(x) = \{r_i | i = 1 \ldots M\}$, where $r_i \leq \tau_k$, $L(r_i) \notin \pi$, and $k = 1, ..., K$.
- Form a labeled rooted forest $F(x) = \{T_1, \ldots T_M\}$, where each T_i is a labeled rooted tree, rooted at r_1 and induced by the descendants of r_i. T_1 is associating with a semantic argument of x.

3.2 Algorithm 2

Let C be a set of categories, $C = \{C_1, C_2, \ldots, C_M\}$, where each C_m is a predefined sense of the ambiguous symbol s_t. The algorithm can be stated as follows:

- Determine contexts of s_t and form $S_t = < s_{t-i} ...s_t... s_{t+j} >$.
- Apply equation (1) to find an optimal category sequence $< c^*_{t-i} \ldots c^*_t \ldots c^*_{t+j} >$ for S_t.
- Assign s_t to C_m if and only if:

$$max\{\#\{c^*_l | c^*_l = C_m \in C, l = t-i, \ldots, t+j\}\}$$

3.3 Algorithm 3

Let S be a sequence of symbols associated with a sentence, $S =< s_1, ..., s_i, ..., s_N >$. Let C be a set of categories, $C = \{C_1, C_2, C_3\}$, where C_1 represents a symbol is inside an noun phrase, C_2 represents a symbol is not in an noun phrase, C_3 represents a symbol starts at a new noun phrase. The algorithm can be stated as follows.

- Apply equation (1) to find an optimal category sequence $< c^*_1, \ldots, c^*_N >$ for S.
- Find $\{B_1, \ldots, B_M\}$, each B_m is a block satisfying the definition of $\mathcal{B}$.
- $\mathcal{B}$ 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 $\mathcal{B}'$ satisfying 1, 2, 3 $\rightarrow \mathcal{B}' \Longleftrightarrow \mathcal{B}$

4 Empirical Results

4.1 Experiments Set Up

We test our model by these three algorithms on data sets: WSJ data from the Peen TreeBank and the PropBank [11], CoNLL-2000 Shared Task Data [10], and data developed by [6] and [1] for WSD. The evaluation metrics we have used are *precision , recall, f-measure* (F_1), and *accuracy* . We have used 10-fold cross validation technique for obtaining an average output.

4.2 Results on the First Algorithm

We have tested our method on the data set developed by [11], specifically, the WSJ section 00 from Penn Treebank and PropBank. A total of 233 trees associates with 233 sentences and 621 verbs. Each verb has an average of three semantic arguments. Hence about 2000 semantic arguments are in total. For each sentence, Peen Treebank provides a corresponding parse tree (with an average accuracy 95%) while PropBank provides corresponding semantic arguments (generated by human labels) of predicates in the sentence.

Corresponding to 621 verbs, we found 621 paths by the equation (1). For each path, a set of roots is found. Then a set of labeled rooted subtrees is formed. The leaves of each tree is associating with a semantic argument of a verb. The test results are shown in Table 1. In average, each time among the $\frac{1}{10}$ semantic arguments that have been classified, about 93% semantic arguments are correctly identified and 7% semantic arguments are classified wrong. By checking these classified instances, we found that our method is very effective in the case of a semantic argument being a sequence of consecutive words. However, if a semantic argument consists of two or more word fragments, separated by some phrases, our algorithm is less effective. The reason is that these phrases are parts of leaves of a tree induced from a root determined by our algorithm. This suggests us that in order to exclude phrases from a semantic argument, we need to develop a method so that a set of subroots is needed to be found. Each of them corresponds to a fragment of a semantic argument. Moreover, other misclassified instances may be generated by errors carried in original syntactic trees.

Table 1. The first algorithm on WSJ data

Files 20,37,49,89	Precision %	Recall %	F-Measure %
Average	92.335	94.1675	93.2512
Standard Deviation	0.6195	0.5174	0.4605

4.3 Results on the Second Algorithm

We have 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 can be found in [6] and [1]. In these data sets, *line* and *interest* are polysemous nouns, *hard* is a polysemous adjective, and *serve* is a polysmous verb. In our experiment, *line* has 6 senses, *serve* has 4 senses, *hard* has 3 senses, *interest* has 3 senses (other 3 senses are omitted due to lack of instances). The test metric that we have used is $accuracy$.

We formed the context of each given target word by including left four open class words and right four open class words combining with the left word and the right word for each of these words. Table 2 shows the test results. In the table, $Mean$ represents the average accuracy, Std represents the standard deviation, $MaxA$ represents the maximum accuracy obtained from tests, and $MinA$ represents the minimum average accuracy obtained obtained from tests.

Table 2. The second algorithm on *line, serve, hard, interest* data

Ambiguous word	Senses	Mean %	Std %	MaxA %	MinA %
Line (n)	6	81.16	1.92	84.50	78.0
	3	85.25	2.13	91.70	81.05
Serve (v)	4	79.80	1.90	82.92	76.88
Hard (adj)	3	82.88	3.10	87.03	78.11
Interest (n)	3	92.10	2.21	95.50	86.00

By observing the experiment results, we found that misclassified instances are primarily generated by the ambiguity of context words in our method. For example in Table 2, comparing with three sense noun $interest$ and three sense noun $line$ (we selected three senses at each time from six senses and examined all twenty combinations), we found that the accuracy of the word $interest$ is almost 9% higher than the one for the word $line$. Moreover, by examining accuracies generated from each combination for the word $line$, we found that some combination ($S_1S_2S_4$) has the highest average accuracy: 91.7% while some combination ($S_1S_3S_5$) has lowest average accuracy: 77.1%. The difference is almost 20%. By carefully checking these misclassified instances, We have learned that if two senses are similar to each other, there are more chances that their contexts consist of same words. As a consequence, the ratio of misclassification increases.

Moreover, by observing the results in Table2, we have noticed that, whether a ambiguous word is a noun, an adjective, or a verb, whether it has three senses, four senses, even six senses, our model has achieved an average of accuracy 80%. This result is very encouraging and surpasses the results published by other researchers [6] and [7]. Moreover, by observing the outputs of two polysemous nouns *line* and *interest*, we found that as number of senses of a polysemous noun increasing, the values of accuracies get reduced. This suggests that nouns with larger number of senses are more difficult to recognize than nouns with small number of senses by our model. Furthermore, by observing the Means in column three, we have noticed that nouns

are relatively easier to identify than adjectives or verbs. By observing the Stds in column four, we have noticed that accuracies generated by our model on adjective data is much divergent than nouns or verbs.

4.4 Results on the Third Algorithm

We have conducted experiments for identifying NP chunks on two kinds of data sets. One is CoNLL-2000 Shared Task data set and the other is WSJ data set from Penn Treebank. On the first data set, three types of symbols are designed to test our model. They are the lexicon of a word, the POS tag of a word, and the lexicon and the POS tag of a word. The results are shown in the table 3. By comparing the results, we have noticed that if the model is built only on the lexical information, it has the lowest performance of F-measure 89.75%. The model's performance improved 3% on F-measure if it is constructed by POS tags. The model achieves the best performance of 95.59% on F-measure if both lexicon and POS tags are included.

The second data set we have used to verify our model is the WSJ data from Penn Treebank: *WSJ 0200 - WSJ 2999*. The main reason for using this data set is that we want to see whether the performance of our model can be improved when it is built on more data. In this experiment, based on the result we have got from the CoNLL-2000 data, only one type of symbols that we have used is lexicon+POS tag. In this case, the training set is seven times larger than the CoNLL-2000 shared task training data set. The test results is shown in Table 3. Note, data inside parentheses in the table represents stand divination.

Compared with the results on these two data sets, we have noticed that the average precision is improved about 2.7% from 95.15% to 97.73% . The average recall is improved about 2.8% from 96.05% to 98.65%. The average F-measure is improved about 2.7% from 95.59% to 98.2% as the training sets expended into the seven times larger. This suggests a tradeoff between sizes of training sets and the performances of our model need to be considered.

Table 3. The third algorithm on $CoNLL-2000$ and WSJ data

Data	Symbol type	Precision %	Recall %	F-measure %
CoNLL	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 (0.19)	98.65 (0.14)	98.18 (0.08)

5 Conclusions

We develop three algorithms for identifying three types of semantic patterns: semantic arguments of a verb, the sense of an ambiguous word, and noun phrases of a sentence, in texts based on a probabilistic graphical model. By this model, a sequence of optimal categories (or a path) for a sequence of symbols (or nodes) is obtained in the way of simple - no need dynamic programming, fast - O(NM), and less memory spaces - O(M) compared with other existed models such as HMMs, CRFs, and $MEMM$s. Moreover, because of the global maximum probability is achieved by finding local maximal probabilities, the ratio of misclassification can be reduced. Performances of these algorithms on numbers of standard data sets demonstrate that our method is effective and efficient.

References

[1] R. Bruce and J. Wiebe. Word-sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–146, 1994.

[2] M. Huang and R. M. Haralick. A probabilistic graphic model for recognizing np chunks from texts. In *The 22nd International Conference on the Computer Processing of Oriental languages*, pages 23–33, 2009.

[3] M. Huang and R. M. Haralick. *Recognizing Patterns in Texts*. River, 2010.

[4] M. Huang and R. M. Haralick. Discovering text patterns by a new graphic model. In *7th International Conference on Machine Learning and Data Mining*, pages 428–442, 2011.

[5] J. Lafferty, A. MaCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[6] C. Leacock, G. Towell, and E. Voorhees. Corpus based statistical sense resolution. In *Proceedings of the workshop on Human Language Technology*, pages 260 – 265, 1993.

[7] E. Levin, M. Sharifi, and J. Ball. Evaluation of utility of lsa for word sense discrimination. In *Preceedings of HLT-NAACL*, pages 77 – 80, 2006.

[8] A. MaCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of 17th International Conf. on Machine Learning*, pages 591–598, 2000.

[9] A. Molina, F. Pla, D. D. S. I. tics, J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. Shallow parsing using specialized hmms. *Journal of Machine Learning Research*, 2:595–613, 2002.

[10] E. F. Tjong and K. Sang. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000*, pages 127–132, 2000.

[11] R. Weischedel, M. Palmer, M. Marcus, and 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.