Self-Teaching Semantic Annotation Method for Knowledge Discovery from Text

Kaiquan Xu
Department of Information Systems, City University of Hong Kong
kaiquan.xu@student.cityu.edu.hk

Stephen Shaoyi Liao
Department of Information Systems, City University of Hong Kong
issliao@cityu.edu.hk

Raymond Y.K. Lau
Department of Information Systems, City University of Hong Kong
raylau@cityu.edu.hk

Lejian Liao
Department of Computer Science, Beijing Institute of Technology liaolj@bit.edu.cn

Heng Tang
Faculty of Business Administration, University of Macao hengtang@umac.mo

Abstract

As much valuable domain knowledge is hidden in enterprises' text repositories (e.g., email archives, digital libraries, etc.), it is desirable to develop effective knowledge management tools to process this unstructured data so as to extract domain knowledge for business decision making. Ontology-based semantic annotation of documents is one of the promising ways for knowledge discovery from text repositories. Existing semantic annotation methods usually require many labeled training examples before they can effectively operate, and this bottleneck holds back the widely applications of these semantic annotation methods. In this paper, we propose a semi-supervised semantic annotation method, self-teaching SVM-struct, which uses fewer labeled examples to improve the annotating performance. The key of the self-teaching method is how to identify the reliably predicted examples for retraining. Two novel confidence measures are developed to estimate prediction confidence. The experimental results show that the prediction performance of our self-teaching semantic annotation method is promising.

1. Introduction

Nowadays, knowledge has become a very key factor for making successful decisions, and most big companies make large investments in Knowledge Management Systems (KMS) and knowledge discovery for decision support. But most data stored in KMS are text documents, and current KMS has limited capabilities for interpreting the text documents. In order to effectively use the valuable text data for decision support, it is necessary to make these texts to be "understandable" by Decision Support Systems (DSS).

Semantic annotating documents using domain ontology is one of the promising ways to implement this object. Semantic annotation formally identifies concepts and relations between concepts in documents, and is intended primarily for use by machines [1]. For example, a semantic annotation might relate "Bei Jing" in a text to an ontology which both identifies it as the abstract concept "City" and links it to the instance "China" of the abstract concept "Country" with the relation of "CapitalOf". Semantic annotation allows for more accurate information retrieval and better interoperability. For example, when searching information about city, "Bei Jing" will be hit, since it is an instance of "City". Interoperability is particularly important for organizations, since their legacy databases exist in different proprietary formats. Semantic annotation with common ontology can facilitate the integration of these heterogeneous data sources. As a result, with semantic annotation, DSS can effectively utilize the valuable knowledge hidden in the text repositories: 1) DSS can "understand" documents to retrieve more accurate information and make better inference for decision support; 2) DSS can exploit vast and heterogeneous legacy data for decision support.

Manual annotating documents needs lots of labor cost and time, so some automatic and semi-automatic methods are proposed. Among them, sequence model based approaches are very effective for semantic annotation, since these models can describe the
dependencies between different parts of information. These sequence models include Hidden Markov Model (HMM) [2], Conditional Random Fields (CRF) [3] and SVM-struct [5]. But these sequence models require many labeled examples as the training examples, and it is generally more complicated and time consuming to annotate the structured training examples for semantic annotation rather than labeling documents for the general classification problems. As a result, sequence models have not been widely used for semantic annotation in practice.

One main contribution of this paper is that a semi-supervised learning method, self-teaching SVM-struct, is proposed for semantic annotation, which only needs fewer labeled examples. The idea of this method is using the reliably predicted examples as the training examples to retrain the SVM-struct model to improve performance. The latest and best sequence model, SVM-struct [5], is adopted here. And according to the characters of SVM-struct, we design two confidence measures to identify the reliably predicted examples for retraining. The experimental results show that this method really improved the performance of semantic annotation.

The rest of the paper is organized as follows: Section 2 introduces some background knowledge of semantic annotation and SVM-struct model. Section 3 describes our self-teaching SVM-struct algorithm and the confidence measures. Section 4 describes our experiment and results. Section 5 introduces the related research in semantic annotation. Finally, we offer concluding remarks and describe future direction of our research work.

2. Background Knowledge

2.1. Semantic Annotation as Sequence Labeling

Semantic annotation can be resolved with the rule induction and classification methods. But recently, sequence model based methods become a promising way for it, since these models enable describing dependencies between concepts, and the dependencies can be utilized to improve the accuracy of the annotation [6]. Here, we use an example to illustrate that semantic annotation is modeled as the sequence labeling problem, and then is resolved with the sequence models.

Bei Jing is the capital of People's Republic of China
City               Country

In the above sentence, "Bei Jing" is an instance of concept "City", and "People's Republic of China" is an instance of "Country". Then its corresponding label sequence is:

Bei Jing is the capital of People's Republic of China
B-City I-City O O O B-Country I-Country I-Country I-Country

Here, the region information is coded by three kinds of labels: "B-concept", "I-concept" and "O". "B-concept" means that the current word is the beginning of an instance of the concept. "I-concept" means that the current word is in an instance of the concept. And "O" indicates that the word is not in an instance of any concept.

After reframing semantic annotation as sequence labeling problem, the task of semantic annotation can be seen as: for a given sentence, finding the right label sequence for it. For the sequence model-based methods, some examples with right label sequences are given as training examples first, and a sequence model is trained based on these examples. Finally, the trained model is used to predicate the label sequences of other unlabeled sentences.

The formal definition of semantic annotation with sequence models is given below:

For a given the ontology Ont, the concept set is defined as
\[ \text{ClassSet} = \{ c_i | c_i \in \text{Ont} \} \], here \( c_i \) is a concept. Then, the label set from this ontology is defined as
\[ \text{LabelSet} = \{ l_i | \exists c | c \in \text{ClassSet} \land \{ l_i = \text{"B"}_c, l_i = \text{"I"}_c, l_i = \text{"O"} \} \} \], that is \( \text{LabelSet} \) consists of the labels with "B" or "I" as the prefixes of concepts. The sequence model-based method for semantic annotation is:

The input space is:
\[ X = \{ x | x_i = w_i w_2 \ldots w_n, w_i \in \text{WordSet} \} \], here \( x \) is a sentence consisting of multiple words;

The output space is:
\[ Y = \{ y | y_i = l_1 l_2 \ldots l_n, l_i \in \text{LabelSet} \} \], here \( y \) is the label sequence.

Given some labeled training examples \( \{(x_i, y_i)\}_{i=1}^{n} \), here \( x_i \in X, y_i \in Y \), a mapping function \( f : X \rightarrow Y \) is learnt using these training examples. For an unlabeled sentence \( x \), \( f(x) \) is the predicted label sequence, which is the semantic annotation for that sentence.

2.2. SVM-struct for Sequence Labeling

There are several sequence models for sequence labeling, such as HMM and CRF. Here, we will adopt SVM-struct, since it leads to better performance than HMM and CRF [7]. The motivation of the SVM-struct algorithm is driven by the idea of the maximum large margin from Support Vector Machine (SVM) [15] for structured output problems, such as sequence labeling and syntactic tree analysis [5]. It tries to find a right
parameter to optimally separate the right structured output from other. The basic idea of SVM-struct is as follows:

Let $X$ and $Y$ are the input and output space, and $x \in X$, $y \in Y$ ($Y$ can be any complex structure), and $\{(x_i, y_i)\}_{i=1}^{n}$ is a set of training examples. First, a linear discriminant function is defined by

$$F(x; y; \theta) \equiv \omega \Psi(x, y),$$

where $\Psi(x, y)$ is the feature representation function and represents the combined features of input $x$ and output $y$. The corresponding mapping function $f$ is defined by the expression

$$f(x; \omega) = \arg \max_{x \in Y} F(x, y; \omega).$$

The SVM-struct method solves the following optimization problem:

$$\min_{\omega} \frac{1}{2} ||\omega||^2 + C \sum_{i=1}^{n} \xi_i \quad \text{s.t.} \forall (1 \leq j \leq n), \forall y : F(x_i, y) - F(x_i, y) \geq \Delta(y, y) - \xi_i,$$

where $\Delta(y, y)$ is the loss function.

For the sequence labeling problem, $X$ is a sentence, and $Y$ is a label sequence. Similar with HMM, the feature representation function can include the combined features of input sentences and labels, and features between adjacent labels. The maximization of $\omega \Psi(x, y)$ over $Y$ can be carried out by dynamic programming (Viterbi algorithm [2]). In the following sections, we exactly utilize the characters of Viterbi algorithm for our novel prediction confidence measures.

### 3. Self-teaching SVM-struct for Semantic Annotation

Although, the previous research shows that SVM-struct leads to promising performance for sequence labeling when compared with other sequence models [7], it suffers from the same weakness of other sequence models, that is, requiring a large number of labeled training examples. In this section, a novel self-teaching method of SVM-struct is proposed to alleviate this partially for semantic annotation.

#### 3.1. Self-Teaching SVM-struct Algorithm

The self-teaching learning is a kind of semi-supervised learning, which uses the reliably predicted unlabeled examples to retrain the model to improve the prediction performance. The process of the self-teaching is: first train a learning model based on small amount of labeled data, and then use the learning model to predict unlabeled examples. Typically, the most confident unlabeled examples, together with their predicted labels, are added into the training example set. The learning model is re-trained and the procedure is repeated until the stop condition is satisfied [8]. The self-teaching SVM-struct algorithm for semantic annotation uses SVM-struct model in the self-teaching process. The algorithm is as follows:

<table>
<thead>
<tr>
<th>Table 1. Self-teaching SVM-struct Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $D_L = {(x_i, y_i)}_{i=1}^{n}$ be the initial labeled training examples.</td>
</tr>
<tr>
<td>Let $D_U = {(x_i, y_i)}_{i=1}^{n}$ be the unlabeled examples.</td>
</tr>
<tr>
<td>1. Train SVM-struct model $M$ on $D_L$.</td>
</tr>
<tr>
<td>2. Use $M$ to predict the label sequence $\hat{y}_j$ for every $x_j$ in $D_U$.</td>
</tr>
<tr>
<td>3. $D_t = D_t \cup {(x_j, \hat{y}_j)}</td>
</tr>
<tr>
<td>4. If $n</td>
</tr>
</tbody>
</table>

The key step 3 in this algorithm means if the confidence of the prediction $\hat{y}_j$ is high enough, $\hat{y}_j$, together with the sentence $x_j$, is added into the training examples set to retrain SVM-struct model. The performance of this algorithm heavily depends on the prediction confidence measure. In the following section, two novel confidence measures are proposed.

#### 3.2. Confidence Measure

In the self-teaching SVM-struct algorithm, the confidence measure of the predicted label sequence plays the key role. A good measure can select the good predicted examples to retrain SVM-struct model, and improve the semantic annotation performance. Otherwise, the poor predicted examples will damage the performance.

For usual generative sequence models (such as HMM), the conditional probability $P(y_j | x_j)$ is a very good measure to identify how probable $\hat{y}_j$ is the right label sequence for $x_j$. But SVM-struct is a discriminative model rather than a generative model, so the conditional probability $P(y_j | x_j)$ is not readily available here. According to the character of SVM-struct, two special confidence measures are described.

#### 3.2.1 Ratio of the Top Two Viterbi Path Costs Measure

In the section 2.2.2, we know that, when
making the prediction for the input $x$, SVM-struct tries to find an output $y$ to maximize $F(x, y; \omega)$, and then this $y$ is the prediction for $x$. For the sequence labeling problem, SVM-struct adopts the Viterbi algorithm [2] to maximize $F(x, y; \omega)$ efficiently. The process is as follows:

Table 2. Viterbi algorithm used in SVM-struct

Let $x = w_1 w_2 \ldots w_i \ldots w_T$ be the input sentence.
Let $l_1, l_2, \ldots, l_i, \ldots l_T \in \text{LabelSet}$ be possible labels.
Let $\delta_i(l_i)$ store the maximal cost of the label path with $l_i$ at the ending position $t$.

Initialization:
$\delta_i(l_i) = \omega \phi(w_i, l_i) >$(I)
where the feature weight vector $\omega$ represents the weights of different features, and the feature representation function $\phi(w_i, l_i)$ represents the combined features of the word $w_i$ and the label $l_i$.

Recursion:
$\delta_{i+1}(l_{i+1}) = \max(\delta_i(l_i) + \omega \phi(w_{i+1}, l_{i+1}) > + \omega \phi(l_{i+1}, l_i))$(II)
t = 1 \ldots T - 1, where the feature representation function $\phi(l_{i+1}, l_i)$ represents the combined features of the adjacent labels, $l_{i+1}$ at the position $t$ and $l_i$ at the position $t + 1$.

Assuming $\delta_i(l_i)$ is the maximal cost, and the labels covered by $\delta_i(l_i)$ compose the predicted label sequence for input $x$.

In picture 1, the path indicated by the solid arrow has the maximal cost, so the labels (the black points) covered by this path are the prediction for the sentence "Bei Jing is the capital of the People's Republic of China".

From the above, we know that the prediction for a sentence is the label sequence with maximal path cost. If a prediction is highly confident, its path cost should have an obvious difference with the second largest path cost. Otherwise, if the difference is small, it is very possible that this prediction is not right. So the ratio of the maximal path cost to the second largest path cost is a good indicator for the prediction confidence. If this ratio exceeds a special threshold, we can think this prediction is confident and can be added into the training example set. Also this ratio can be calculated efficiently with the Viterbi algorithm for the maximal path cost and second largest path cost.

3.2.2 Annotated Semantic Concepts Confidence Measure. Although the ratio of the top two path costs can measure the confidence of the predicted label sequence, it does not differentiate the semantic concept labels ("B-Concept" and "I-Concept") from the blank label ("O"), and handle them totally equally. So this measure does not consider the confidence of the annotated semantic concepts, which should be the focus points in semantic annotation. We propose another confidence measure, the Annotated Semantic Concepts Confidence Measure. The intuition of this measure is that assuming the semantic concepts annotated in the maximal cost path are right, and these right concepts are labeled as other concepts in another path, this path cost will decrease greatly. So we use the ratio of the maximal path cost to the maximal cost of the path, which bypasses the semantic concepts annotated in the maximal cost path, as the second confidence measure.

\[\begin{array}{c}
B-Country \\
O \\
B-City \\
I-City \\
I-Country \\
\end{array}\]

Picture 1. An Example of the Prediction
In picture 2, the concept labels covered by the maximal cost path in picture 1 are circled, which the label sequence must not pass when searching the maximal cost path bypassing the annotated semantic concepts. In this example, if the searched path is the one indicated by the dashed arrow, the prediction confidence using this measure is the ratio of the cost of the black arrow path in picture 1 to the cost of the dashed arrow path in picture 2.

For calculating the maximal cost of the path bypassing the special semantic concept labels, some constrains are added into the Viterbi algorithm, like \[ C \neq t \]. For the above example, \( C \) should be \( \{ s_i \neq B \_City, s_i \neq I \_City,... \} \). After adding some constrains, the formulas (I) and (II) in table 2 are modified separately as follows:

\[
\delta_i(l_i) = \begin{cases} 0 & \text{if } (l_i = l'_i) \\ <a, \phi(w_i, l_i)> & \text{otherwise} \end{cases}
\]

\[
\delta_{\omega_i}(l_i) = \begin{cases} 0 & \text{if } (l_i = l'_{i-1}) \\ \max_j(\delta_j(l'_i) + <a, \phi(w_{i-1}, l_i)>) > <a, \phi(l'_{i-1}, l_i)>) & \text{otherwise} \end{cases}
\]

So this confidence measure can also be computing efficiently.

4. Experimental Evaluation

4.1. Experiment Design

In order to evaluate the performance of the self-teaching SVM-struct for semantic annotation, a controlled experiment was performed. The results were compared with the sequence model based methods.

The GENIA benchmark corpus [9] was used for this experiment. The GENIA corpus is an annotated corpus for the biology domain. Some terms are annotated with the GENIA concept labels. The ontology for this experiment comes from the "organic" branch of the GENIA ontology and contains 27 classes with 4 levels and 23 leaf node classes. After some preprocessing, 1144 sentences and the related label sequences were obtained. Two types of features associated with input sentences were used: 1) word feature \( w_i \), \( k \) is the relative position from the current word, a negative value represents the preceding word, and a positive value represents the following word. 2) Part-Of-Speech (POS) tag of the current word \( pos \).

The sequence model based method and the self-teaching SVM-struct method with the two confidence measures were applied separately in this experiment. The three metrics widely used in the information retrieval field, precision, recall, and f-score, were adopted in this experiment. Precision means the ratio of the right predicted semantic concepts to all predicted semantic concepts by the algorithm, recall means the ratio of the right predicted semantic concepts by the algorithm to all original semantic concepts, and f-score is a mixture of precision and recall, \( f-score = (2 \cdot \text{precision} \cdot \text{recall})/(\text{precision} + \text{recall}) \). The SVM-struct [10] software packages were revised in this experiment.

The experiment contained two settings. In setting 1, 917 sentences (about 24,000 words) and their labels were as the training examples, and 227 sentences (about 6,000 words) were as the testing examples. In setting 2, 193 sentences (about 5,000 words) and their labels were as the training examples, and 227 sentences (about 6,000 words) were as the testing examples.

4.2. Experimental Results

The initial experimental results are reported in table 3, 4 and figure 3, 4 (confidence measure 1 is the ratio of the top two path costs; confidence measure 2 is the annotated semantic concepts confidence measure).

<table>
<thead>
<tr>
<th>Method</th>
<th>f-score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-struct</td>
<td>54.46%</td>
<td>60.70%</td>
<td>56.64%</td>
</tr>
<tr>
<td>Self-teaching SVM-struct</td>
<td>54.38%</td>
<td>60.56%</td>
<td>49.57%</td>
</tr>
<tr>
<td>with confidence measure 1</td>
<td>55.04%</td>
<td>61.20%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Self-teaching SVM-struct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with confidence measure 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>f-score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-struct</td>
<td>36.52%</td>
<td>43.84%</td>
<td>31.70%</td>
</tr>
</tbody>
</table>

Table 3. The Experiment Result in Setting 1

Table 4. The Experiment Result in Setting 2
### 5. Related Work

Many research efforts have been done for semantic annotation. However, most of the previous work uses classification methods or sequence model based methods, and few of them tried to use fewer labeled training examples to improve the performance. The previous work can be categorized as the follows:

1. **Rule Induction Method.** Rule induction is employed for the semantic annotation. For example, [4] can learn annotation rules from the training data. Although rule induction has better precision, the recall is not very high.

2. **Classification Method.** The method views semantic annotation as a problem of classification. It learns a classifier from training examples to detect the types of the test examples. For example, SCORE Enhancement Engine (SEE) supports web page annotation by using classification model [11].

3. **Sequence Model Based Method.** [13] utilized HMM in semantic annotation. But HMM needs lots of training examples to enumerate all possible observation sequences.

### 6. Conclusions and Future Work

In this paper, a novel self-teaching SVM-struct model is proposed to improve the performance of semantic annotation with fewer labeled examples. Especially, two prediction confidence measures are described for the self-teaching SVM-struct. The experimental results show this method achieves promising performances.

In the future, more features from text, such as prefix, suffix, verb, syntactic relations, etc., will be added into the models to improve the precision. In addition, the co-training method is also an optional way to increase the performance, and we will explore the co-training between CRF and SVM-struct models for semantic annotation.

### 7. References:


