A Novel Framework of Training Hidden Markov Support Vector Machines from Lightly-Annotated Data

Deyu Zhou
School of Computer Science and Engineering
Southeast University, Nanjing, China
d.zhou@seu.edu.cn

Yulan He
Knowledge Media Institute
Open University, UK
y.he@open.ac.uk

ABSTRACT

Natural language understanding (NLU) aims to map sentences to their semantic mean representations. Statistical approaches to NLU normally require fully-annotated training data where each sentence is paired with its word-level semantic annotations. In this paper, we propose a novel learning framework which trains the Hidden Markov Support Vector Machines without the use of expensive fully-annotated data. In particular, our learning approach takes as input a training set of sentences labeled with abstract semantic annotations encoding underlying embedded structural relations and automatically induces derivation rules that map sentences to their semantic meaning representations. The proposed approach has been tested on the DARPA Communicator Data and achieved 93.18% in F-measure, which outperforms the previously proposed approaches of training the hidden vector state model or conditional random fields from unaligned data, with a relative error reduction rate of 43.3% and 10.6% being achieved.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithms, Experimentation

Keywords

Hidden Markov support vector machines (HM-SVMs), Natural language understanding, Semantic parsing

1. INTRODUCTION

The natural language understanding problem can be considered as a mapping problem where the aim is to map a sentence to its semantic meaning representation (or abstract semantic annotation) such as the one shown below.

CIKM'11, October 24–28, 2011, Glasgow, UK.

I want to return to Dallas on Thursday.
`RETURN(TOLOC(CITY(Dallas)) ON(DATE(Thursday)))`

This is a *structured classification* task which predicts output labels (semantic tag or concept sequences) C from input sentences S where the output labels have rich internal structures. It is a highly challenging problem because the derivation from each sentence to its abstract semantic annotation is not annotated in the training data and is considered hidden. Although a hierarchical hidden state structure could be used to model embedded structural context in sentences, such as the Hidden Vector State (HVS) model [4], which learns a probabilistic push-down automaton, it cannot include a large number of correlated lexical or syntactic features in input sentences, and it cannot handle any arbitrary embedded relations since it only supports right-branching semantic structures. Other approaches learn semantic parsers that map natural language sentences into a formal meaning representation such as lambda calculus or first-order logic [6, 2]. However these systems either require a hand-built, ambiguous combinatory categorial grammar template to learn a probabilistic semantic parser [6], or assume the existence of an unambiguous, context-free grammar of the target meaning representations [2].

Conditional Random Fields (CRFs) have been extensively studied for sequence labeling. Most applications require the availability of fully annotated data, i.e., an explicit alignment of sentence and word-level labels. Mann and McCallum [5] use labeled features instead of fully labeled instances to train linear-chain CRFs. Generalized expectation criteria are used to express a preference for parameter settings in which the model's distribution on unlabeled data matches a target distribution. They tested their approach on the classified advertisements data set (Classified) [3] and achieved 68.3% accuracy with only labeled features. Zhou and He [7] proposed an iterative learning approach based on expectation maximization (EM) to train the CRFs from abstract semantic annotations. They achieved 92% in F-measure on the DARPA Communicator Data.

In this paper, we propose a novel learning approach to train the HM-SVMs from unaligned data. It first computes expectations using initial model parameters. Parsing results are then filtered based on a measure describing the level of agreement with the sentence abstract semantic annotations and fed into model learning using the cutting-plane algorithm. With the re-estimated parameters, the learning of HM-SVMs goes to the next iteration until no more improvements could be achieved. The rest of this paper is organized as follows. Section 2 introduces HM-SVMs. The pro-

posed learning procedure to train HM-SVMs from abstract semantic annotations is presented in Section 3. Experimental setup and results are discussed in Section 4. Finally, Section 5 concludes the paper.

2. HIDDEN MARKOV SUPPORT VECTOR MACHINES (HM-SVMS)

Given a set of training data $(S_i, C_i), i = 1, \ldots N$, to learn a function that assigns to a sequence of words $S = (s^1 \cdots s^T), s^i \in \mathbf{s}, i = 1, \ldots T$, a sequence of semantic concepts or tags $C = (c^1 c^2 \ldots c^T), c^i \in \mathbf{c}, i = 1, \ldots T$, a common approach is to find a discriminant function $F : \mathcal{S} \times \mathcal{C} \rightarrow \mathbb{R}$ that assigns a score to every input $S \in \mathcal{S}$ and every semantic tag sequence $C \in \mathcal{C}$. In order to obtain a prediction $f(S) \in \mathcal{C}$, the function is maximized with respect to $f(S) = \underset{C \in \mathcal{C}}{\operatorname{argmax}} F(S, C)$.

In particular, the function $F(S, C)$ is assumed to be linear in some combined feature representation of S and C in HM-SVMs [1], $F(S, C) := \langle w, \Phi(S, C) \rangle$. The parameters w are adjusted so that the true semantic tag sequence C_i scores higher than all other tag sequences $C \in \mathcal{C}_i := \mathcal{C} \backslash C_i$ with a large margin. To achieve the goal, the following optimization problem is solved instead:

$$\begin{aligned} \min_{\xi_i \in \mathbb{R}, w \in \mathcal{F}} \quad & Cons \sum_i \xi_i + \frac{1}{2}\|w\|^2 \qquad (1) \\ \text{s.t.} \quad & \langle w, \Phi(S, C_i) \rangle - \langle w, \Phi(S, C) \rangle \geq 1 - \xi_i, \\ & \forall i = 1, \ldots N \text{ and } C \in \mathcal{C} \backslash C_i \end{aligned}$$

where ξ_i is non-negative slack variables allowing one to increase the global margin by paying a local penalty on some outlying examples, and $Cons$ dictates the desired trade off between margin size and outliers. To solve Equation 1, the dual of the equation is solved instead. The solution $\hat{w}$ can be written as

$$\hat{w} = \sum_{i=1}^{N} \sum_{C \in \mathcal{C}} \alpha_i(C) \Phi(S_i, C), \qquad (2)$$

where $\alpha_i(C)$ is the Lagrange multiplier of the constraint associated with example i and C_i.

3. TRAINING HM-SVMS FROM ABSTRACT SEMANTIC ANNOTATIONS

To train HM-SVMs from abstract semantic annotations, we extended the use of expectation maximization (EM) algorithm to estimate model parameters. The EM algorithm is an efficient iterative procedure to compute the maximum likelihood (ML) estimate in the presence of missing or hidden data. The EM algorithm is divided into two steps. In the E-step, the missing data are estimated given the observed data and current estimate of model parameters. In the M-step, the likelihood function is maximized under the assumption that the missing data are known. We summarize the procedure of learning HM-SVMs from abstract semantic annotations in Figure 3. The details of each step are given in the subsequent sections.

3.1 Initialization

Given a sentence labeled with an abstract semantic annotation as shown in Table 1, we first expand the annotation to the flattened semantic tag sequence as in Table 1(a). In order to cater for irrelevant input words, a DUMMY tag is introduced in the preterminal position. Hence, the flattened semantic tag sequence is finally expanded to the semantic tag sequence as in Table 1(b).

Input:	A set of sentences $\mathbf{S} = \{S_i, i = 1, \ldots, N\}$ and their corresponding semantic annotations $\mathbf{A} = \{A_i, i = 1, \ldots, N\}$
Output:	The trained HM-SVMs with parameters Θ
Procedure:	
1.	**Initialization**: For each A_i, expand the annotation to the flattened semantic tag sequence C_i. Estimate the initial model parameters Θ_0 using $\mathbf{S}$ and the flattened semantic tag sequences $\mathbf{C} = \{C_i, i = 1, \ldots, N\}$. Set $\Theta = \Theta_0$.
2.	**Expectation**: For each S_i, generate the semantic tag sequence $\hat{C}_i$ using the HM-SVMs with the current model parameters Θ.
3.	**Filtering**: For the generated semantic tag sequences $\hat{\mathbf{C}} = \{\hat{C}_i, i = 1, \ldots, N\}$, filter $\hat{\mathbf{C}}$ based on a score function which measures the agreement of the generated semantic tag sequences with the actual flattened semantic tag sequences $\mathbf{C}$.
4.	**Maximization**: Re-estimate model parameters Θ' using the filtered $\hat{\mathbf{C}}$ and set $\Theta = \Theta'$.
5.	If converged, Output Θ. Else go to Step 2.

Figure 1: Procedure of learning HM-SVMs from abstract semantic annotations.

Table 1: Abstract semantic annotation and its flattened semantic tag sequence.

Sentence:	I want to return to Dallas on Thursday.
Annotation:	`RETURN(TOLOC(CITY(Dallas))` `ON(DATE(Thursday)))`
(a) Flattened semantic tag list:	
`RETURN RETURN+TOLOC RETURN+TOLOC+CITY(Dallas)`	
`RETURN+ON RETURN+ON+DATE(Thursday)`	
(b) Expanded semantic tag list:	
`RETURN RETURN+DUMMY RETURN+TOLOC RETURN+TOLOC+DUMMY`	
`RETURN+TOLOC+CITY(Dallas)`	
`RETURN+TOLOC+CITY(Dallas)+DUMMY`	
`RETURN+ON RETURN+ON+DUMMY RETURN+ON+DATE(Thursday)`	
`RETURN+ON+DATE(Thursday)+DUMMY`	

3.2 Expectation

We first need to calculate the most likely semantic tag sequence $\hat{C}$ for each sentence $S = (s^1 \cdots s^T)$, $\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}} F(S, C)$ where $F : \mathcal{S} \times \mathcal{C} \rightarrow \mathbb{R}$ is a discriminant function and can be decomposed into two components, $F(S, C) = F_1(S, C) +$

$F_2(S, C)$, where

$$F_1(S, C) = \sum_{\sigma \in \mathbf{c}, \tau \in \mathbf{c}} \delta(\sigma, \tau) \sum_{l=1}^{T} [[c^{l-1} = \sigma \wedge c^l = \tau]] \quad (3)$$

$$F_2(S, C) = \sum_{\sigma \in \mathbf{c}} \sum_{l=1}^{T} \gamma(s^l, \sigma)[[c^l = \sigma]] \quad (4)$$

Here, $\delta(\sigma, \tau)$ is considered as the co-efficient for the transition from state (or semantic tag) σ to state τ while $\gamma(s^l, \sigma)$ can be treated as the co-efficient for the emission of word s^l from state σ. They are defined as follows,

$$\delta(\sigma, \tau) = \sum_{i, \overline{C}} \alpha_i(\overline{C}) \sum_{m=1}^{|\overline{C}|} [[\overline{c}^{m-1} = \sigma \wedge \overline{c}^m = \tau]] \quad (5)$$

$$\gamma(s^l, \sigma) = \sum_{i,m} \sum_{C} [[c^m = \sigma]] \alpha_i(C) k(s^l, s_i^m) \quad (6)$$

where $k(s^l, s_i^m) = \langle \Psi(s^l), \Psi(s_i^m) \rangle$ describes the similarity of the input patterns Ψ between word s^l and word s_i^m, the mth word in the training example i, $\alpha_i(C)$ is a set of dual parameters or Lagrange multiplier of the constraint associated with example i and semantic tag sequence C as in Equation 2. Using the results derived in Equations 5 and 6, Viterbi decoding can be performed to generate the best semantic tag sequence.

To incorporate the constraints as defined in the abstract semantic annotations, the values of $\delta(\sigma, \tau)$ and $\gamma(s^l, \sigma)$ are modified for each sentence,

$$\delta(\sigma, \tau) = \begin{cases} 0, \text{ when } g(\sigma, \tau) = 1 \\ \sum_{i, \overline{C}} \alpha_i(\overline{C}) \sum_m [[\overline{c}^{m-1} = \sigma \wedge \overline{c}^m = \tau]], \text{otherwise} \end{cases}$$

$$\gamma(s^l, \sigma) = \begin{cases} 0, \text{ when } h(\sigma, s^l) = 1 \\ \sum_{i,m} \sum_C [[c^m = \sigma]] \alpha_i(C) k(s^l, s_i^m), \text{otherwise} \end{cases}$$

where $g(\sigma, \tau)$ and $h(\sigma, s^l)$ are defined as follows,

$$g(\sigma, \tau) = \begin{cases} 1, & \tau \text{ is not in the allowable semantic tag list} \\ 0, & \text{otherwise} \end{cases}$$

$$h(\sigma, s^l) = \begin{cases} 1, & \sigma \text{ is not of class type and } s^l \text{ is of class type} \\ 0, & \text{otherwise} \end{cases}$$

$g(\sigma, \tau)$ and $h(\sigma, s^l)$ in fact encodes the two constraints implied from abstract annotations. Firstly, state transitions are only allowed if both incoming and outgoing states are listed in the semantic annotation defined for the sentence. Secondly, if there is a lexical item attached to a preterminal tag of a flattened semantic tag, that semantic tag must appear bound to that lexical item in the training annotation. For example, in the annotation shown in Table 1, 'Dallas' belongs to a lexical class 'CITY'. Hence, it can only be tagged with semantic tags containing a preterminal tag 'CITY'.

3.3 Filtering

For each sentence, the semantic tag sequences generated in the *Expectation* step are further processed based on a measure on the agreement of the semantic tag sequence $T = \{t_1, t_2, ..., t_n\}$ with its corresponding abstract semantic annotation A. The score of T is defined as

$$Score(T) = 2 * \frac{S_{recall} * S_{precision}}{S_{recall} + S_{precision}}, \quad (7)$$

where $S_{precision} = N_r/n$,$S_{Recall} = N_r/p$. Here, N_r is the number of the semantic tags in T which also occur in A, n is the number of semantic tags in T, and p is the number of semantic tags in the flattened semantic tag sequence for A. The score is similar to the F-measure which is the harmonic mean of precision and recall. It essentially measures the agreement of the generated semantic tag sequence with the abstract semantic annotation. We filter out sentences with their score below certain predefined threshold and the remaining sentences together with their generated semantic tag sequences are fed into the next *Maximization* step. In our experiments, we empirically set the threshold to 0.1.

3.4 Maximization

Given the filtered training examples from the *Filtering* step, the parameters w are adjusted so that the true semantic tag sequence C_i scores higher than all the other tag sequences $C \in \mathcal{C}_i := \mathcal{C} \backslash C_i$ with a large margin. To achieve the goal, the optimization problem as stated in Equation 1 is solved using an online learning approach as described in [1]. In short, it works as follows, a pattern sequence S_i is presented and the optimal semantic tag sequence $\hat{C}_i = f(S_i)$ is computed by employing Viterbi decoding. If $\hat{C}_i$ is correct, no update is performed. Otherwise, the weight vector w is updated based on the difference from the true semantic tag sequence $\Delta\Phi = \Phi(S_i, \hat{C}_i) - \Phi(S_i, C_i)$.

4. EXPERIMENTS

Experiments have been conducted on the DARPA Communicator data[1] which were collected in 461 days. From these, 46 days were randomly selected for use as test set data and the remainder were used for training. After cleaning up the data, the training set consist of 12702 utterances while the test set contains 1178 utterances.

The abstract semantic annotations used for training only list a set of valid semantic tags and the dominance relationships between them without considering the actual realized semantic tag sequence or attempting to identify explicit word/concept pairs. Thus, it avoids the need for expensive tree-bank style annotations. For example, for the sentence "Show me flights from Boston to New York", the abstract annotation would be
`FLIGHT(FROMLOC(CITY) TOLOC(CITY)).`

To evaluate the performance of the model, a reference frame structure was derived for every test set sentence consisting of slot/value pairs. An example of a reference frame is:

Show me flights from Boston to New York.	
Frame:	FLIGHT
Slots:	FROMLOC.CITY = Boston
	TOLOC.CITY = New York

Performance was then measured in terms of F-measure on slot/value pairs, which combines the precision (P) and recall (R) values with equal weight and is defined as $F = (P + R)/2PR$. We modified the open source SVM^{hmm}[2] to train and test the HM-SVMs on abstract annotations.

[1] `http://www.bltek.com/spoken-dialog-systems/cu-communicator.html`

[2] `http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html`

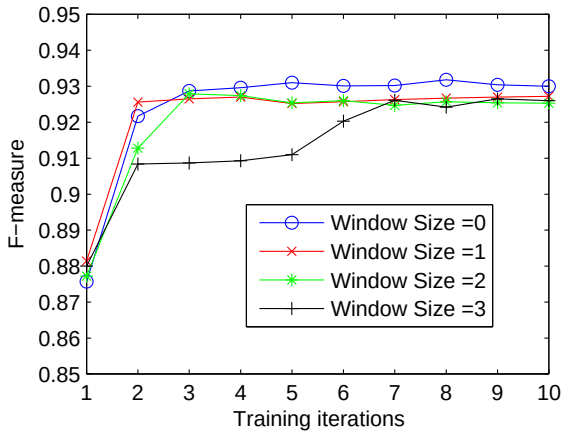

Figure 2: Comparison of performance on models learned with feature sets chosen based on different window sizes.

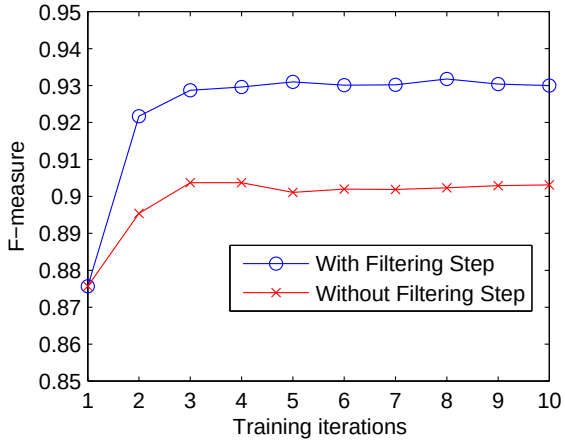

Figure 3: Comparisons of performance with or without the *Filtering* stage.

We employed word features (such as current word, previous word, and next word, etc) and part-of-speech (POS) features (such as current POS tag, previous one, and next one, etc) for training. To explore the impact of the choices of features, we explored with feature sets comprising of words or POS tags occurring before or after the current word within some predefined window size. Figure 2 shows the performance of our proposed approach with the window size varying between 0 and 3. Surprisingly, the model learned with feature set chosen by setting window size 0 gives the best overall performance. Varying window size between 1 and 3 only impacts the convergence rate and does not lead to any performance difference at the end of the learning procedure.

In a second set of experiments, we compare the performance with or without the *Filtering* step as discussed in Section 3.3. Figure 3 shows that the *Filtering* step is indeed crucial as it boosted the performance by nearly 3%.

We compare the performance of HM-SVMs with HVS and CRFs, all trained on abstract semantic annotations. The HVS model [4] was previously proposed based on the hypothesis that a suitably constrained hierarchical model may be trainable without treebank data whilst simultaneously retaining sufficient ability to capture the hierarchical structure need to robustly extract task domain semantics. Such a constrained hierarchical model can be conveniently implemented using the HVS model which extends the HMM model by expanding each state to encode the stack of a push-down automaton. While it is hard to incorporate arbitrary input features to HVS learning, both HM-SVMs and CRFs have the capability to deal with overlapping features. A learning approach of training CRFs from abstract annotations was previously proposed in [7]. Table 2 shows that HM-SVMs outperforms both HVS and CRFs with a relative error reduction of 43.3% and 10.6% being achieved respectively. The superior performance of HM-SVMs over CRFs shows the advantage of HM-SVMs on learning non-linear discriminant functions via kernel functions.

Table 2: Overall comparison with other models.

Measurement	HVS	CRFs	HM-SVMs
Recall (%)	87.81	92.27	92.04
Precision (%)	88.13	92.48	94.36
F-measure (%)	87.97	92.37	93.18

5. CONCLUSIONS

In this paper, we proposed an effective learning approach which can train HM-SVMs without the expensive annotated data. It takes as input a training set of sentences labeled with abstract semantic annotations encoding underlying embedded structural relations and automatically induces derivation rules that map sentences to semantic meaning representation. We evaluated the performance of our proposed learning approach on the DARPA Communicator Data and showed that it outperforms two other models, HVS and CRFs, also trained on abstract annotations.

6. REFERENCES

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *ICML*, pages 3–10, 2003.

[2] R. Ge and R. Mooney. Learning a Compositional Semantic Parser using an Existing Syntactic Parser. In *ACL*, pages 611–619, 2009.

[3] T. Grenager, D. Klein, and C. D. Manning. Unsupervised learning of field segmentation models for information extraction. In *ACL*, pages 371–378, 2005.

[4] Y. He and S. Young. Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19(1):85–106, 2005.

[5] G. S. Mann and A. Mccallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*, pages 870–878, 2008.

[6] L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.

[7] D. Zhou and Y. He. Learning conditional random fields from unaligned data for natural language understanding. In *ECIR*, pages 283–288, 2011.