In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 107–110, Lisbon, Portugal, 2000.

Shallow Parsing by Inferencing with Classifiers*

Vasin Punyakanok and **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{punyakan, danr}@cs.uiuc.edu

Abstract

We study the problem of identifying phrase structure. We formalize it as the problem of combining the outcomes of several different classifiers in a way that provides a coherent inference that satisfies some constraints, and develop two general approaches for it. The first is a Markovian approach that extends standard HMMs to allow the use of a rich observations structure and of general classifiers to model state-observation dependencies. The second is an extension of constraint satisfaction formalisms. We also develop efficient algorithms under both models and study them experimentally in the context of shallow parsing.[1]

1 Identifying Phrase Structure

The problem of identifying phrase structure can be formalized as follows. Given an input string $O =< o_1, o_2, \ldots, o_n >$, a *phrase* is a substring of consecutive input symbols $o_i, o_{i+1}, \ldots, o_j$. Some external mechanism is assumed to consistently (or stochastically) annotate substrings as phrases[2]. Our goal is to come up with a mechanism that, given an input string, identifies the phrases in this string. this is a fundamental task with applications in natural language (Church, 1988; Ramshaw and Marcus, 1995; Muñoz et al., 1999; Cardie and Pierce, 1998).

The identification mechanism works by using classifiers that process the input string and recognize in the input string local signals which are indicative to the existence of a phrase. Local signals can indicate that an input symbol o is ***i***nside or ***o***utside a phrase (IO modeling) or they can indicate that an input symbol o ***o***pens or ***c***loses a phrase (the OC modeling) or some combination of the two. In any case, the local signals can be combined to determine the phrases in the input string. This process, however, needs to satisfy some constraints for the resulting set of phrases to be legitimate. Several types of constraints, such as length and order can be formalized and incorporated into the mechanisms studied here. For simplicity, we focus only on the most basic and common constraint - we assume that phrases do not overlap.

The goal is thus two-fold: to learn classifiers that recognize the local signals and to combine these in a ways that respects the constraints.

2 Markov Modeling

HMM is a probabilistic finite state automaton used to model the probabilistic generation of sequential processes. The model consists of a finite set $\mathcal{S}$ of states, a set $\mathcal{O}$ of observations, an initial state distribution $P_1(s)$, a state-transition distribution $P(s|s')$ for $s, s' \in \mathcal{S}$ and an observation distribution $P(o|s)$ for $o \in \mathcal{O}$ and $s \in \mathcal{S}$.[3]

In a supervised learning task, an observation sequence $O =< o_1, o_2, \ldots o_n >$ is supervised by a corresponding state sequence $S =< s_1, s_2, \ldots s_n >$. The supervision can also be supplied, as described in Sec. 1, using the local signals. Constraints can be incorporated into the HMM by constraining the state transition probability distribution $P(s|s')$. For example, set $P(s|s') = 0$ for all s, s' such that the transition from s' to s is not allowed.

* This research is supported by NSF grants IIS-9801638, SBR-9873450 and IIS-9984168.

[1] Full version is in (Punyakanok and Roth, 2000).

[2] We assume here a single type of phrase, and thus each input symbol is either in a phrase or outside it. All the methods we discuss can be extended to deal with several kinds of phrases in a string, including different kinds of phrases and embedded phrases.

[3] See (Rabiner, 1989) for a comprehensive tutorial.

Combining HMM and classifiers (artificial neural networks) has been exploited in speech recognition (Morgan and Bourlard, 1995), however, with some differences from this work.

2.1 HMM with Classifiers

To recover the most likely state sequence in HMM, we wish to estimate all the required probability distributions. As in Sec. 1 we assume to have local signals that indicate the state. That is, we are given classifiers with states as their outcomes. Formally, we assume that $P_t(s|o)$ is given where t is the time step in the sequence. In order to use this information in the HMM framework, we compute

$$P_t(o|s) = P_t(s|o)P_t(o)/P_t(s). \tag{1}$$

instead of observing the conditional probability $P_t(o|s)$ directly from training data, we compute it from the classifiers' output. $P_t(s)$ can be calculated by $P_t(s) = \sum_{s' \in \mathcal{S}} P(s|s')P_{t-1}(s')$ where $P_1(s)$ and $P(s|s')$ are the two required distribution for the HMM. For each t, we can treat $P_t(o|s)$ in Eq. 1 as a constant η_t because our goal is only to find the most likely sequence of states for given observations which are the same for all compared sequences. Therefore, to compute the most likely sequence, standard dynamic programming (Viterbi) can still be applied.

2.2 Projection based Markov Model

In HMMs, observations are allowed to depend only on the current state and long term dependencies are not modeled. Equivalently, from the constraint point of view, the constraint structure is restricted by having a stationary probability distribution of a state given the previous one. We attempt to relax this by allowing the distribution of a state to depend, in addition to the previous state, on the observation. Formally, we make the independence assumption:

$$\begin{aligned} P(s_t|s_{t-1}, s_{t-2}, \ldots, s_1, o_t, o_{t-1}, \ldots, o_1) \\ = P(s_t|s_{t-1}, o_t). \end{aligned} \tag{2}$$

Thus, we can find the most likely state sequence S given O by maximizing

$$\begin{aligned} P(S|O) &= \prod_{t=2}^{n}[P(s_t|s_1, \ldots, s_{t-1}, O)]P_1(s_1|O) \\ &= \prod_{t=2}^{n}[P(s_t|s_{t-1}, o_t)]P_1(s_1|o_1). \end{aligned} \tag{3}$$

Hence, this model generalizes the standard HMM by combining the state-transition probability and the observation probability into one function. The most likely state sequence can still be recovered using the dynamic programming algorithm over the Eq.3.

In this model, the classifiers' decisions are incorporated in the terms $P(s|s', o)$ and $P_1(s|o)$. In learning these classifiers we project $P(s|s', o)$ to many functions $P_{s'}(s|o)$ according to the previous states s'. A similar approach has been developed recently in the context of maximum entropy classifiers in (McCallum et al., 2000).

3 Constraint Satisfaction with Classifiers

The approach is based on an extension of the Boolean constraint satisfaction formalism (Mackworth, 1992) to handle variables that are outcomes of classifiers. As before, we assume an observed string $O =< o_1, o_2, \ldots o_n >$ and local classifiers that, w.l.o.g., take two distinct values, one indicating the ***o***penning a phrase and a second indicating ***c***losing it (OC modeling). The classifiers provide their outputs in terms of the probability $P(o)$ and $P(c)$, given the observation.

To formalize this, let E be the set of all possible phrases. All the non-overlapping constraints can be encoded in: $f = \bigwedge_{e_i \ overlaps \ e_j} (\neg e_i \vee \neg e_j)$. Each solution to this formulae corresponds to a legitimate set of phrases.

Our problem, however, is not simply to find an assignment $\tau : E \rightarrow \{0, 1\}$ that satisfies f but rather to optimize some criterion. Hence, we associate a cost function $c : E \rightarrow [0, 1]$ with each variable, and then find a solution τ of f of minimum cost, $c(\tau) = \sum_{i=1}^{n} \tau(e_i)c(e_i)$. In phrase identification, the solution to the optimization problem corresponds to a shortest path in a directed acyclic graph constructed on the observation symbols, with legitimate phrases (the variables in E) as its edges and their costs as the weights. Each path in this graph corresponds to a satisfying assignment and the shortest path is the optimal solution.

A natural cost function is to use the classifiers probabilities $P(o)$ and $P(c)$ and define, for a phrase $e = (o, c)$, $c(e) = 1 - P(o)P(c)$ which means that the error in selecting e is the error in selecting either o or c, and allowing those to overlap[4]. The constant in $1 - P(o)P(c)$ biases the minimization to prefers selecting a few phrases, possibly no phrase, so instead we minimize $-P(o)P(c)$.

4 Shallow Parsing

The above mentioned approaches are evaluated on shallow parsing tasks. We use the OC modeling and learn two classifiers; one predicting whether there should be a ***o***pen in location t or not, and the other whether there should a ***c***lose in location t or not. For technical reasons it is easier to keep track of the constraints if the cases $\neg \boldsymbol{o}$ and $\neg \boldsymbol{c}$ are separated according to whether we are inside or outside a phrase. Consequently, each classifier may output three possible outcomes **O**, **nOi**, **nOo** (open, not open inside, not open outside) and **C**, **nCi**, **nCo**, resp. The state-transition diagram in figure 1 captures the order constraints. Our modeling of the problem is a modification of our earlier work on this topic that has been found to be quite successful compared to other learning methods attempted on this problem (Muñoz et al., 1999) and in particular, better than the IO modeling of the problem (Muñoz et al., 1999).

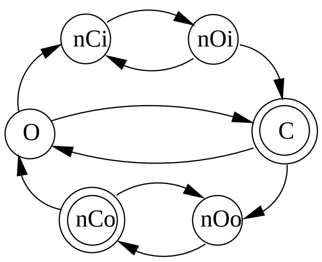

Figure 1: State-transition diagramfor the phrase recognition problem.

The classifier we use to learn the states as a function of the observations is SNoW (Roth, 1998; Carleson et al., 1999), a multi-class classifier that is specifically tailored for large scale learning tasks. The SNoW learning architecture learns a sparse network of linear functions, in which the targets (states, in this case) are represented as linear functions over a common feature space. Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation value of the target classes in this case. The activation value itself is computed using a sigmoid function over the linear sum. In this case, instead, we normalize the activation levels of all targets to sum to 1 and output the outcomes for all targets (states). We verified experimentally on the training data that the output for each state is indeed a distribution function and can be used in further processing as $P(s|o)$ (details omitted).

5 Experiments

We experimented both with base noun phrases (NP) and subject-verb patterns (SV) and show results for two different representations of the observations (that is, different feature sets for the classifiers) - part of speech (POS) tags only and POS with additional lexical information (words). The data sets used are the standard data sets for this problem (Ramshaw and Marcus, 1995; Argamon et al., 1999; Muñoz et al., 1999; Tjong Kim Sang and Veenstra, 1999) taken from the Wall Street Journal corpus in the Penn Treebank (Marcus et al., 1993).

For each model we study three different classifiers. The *simple* classifier corresponds to the standard HMM in which $P(o|s)$ is estimated directly from the data. The NB (naive Bayes) and SNoW classifiers use the same feature set, conjunctions of size 3 of POS tags (+ words) in a window of size 6 around the target word.

The first important observation is that the SV task is significantly more difficult than the NP task. This is consistent for all models and all features sets. When comparing between different models and features sets, it is clear that the simple HMM formalism is not competitive with the other two models. What is interesting here is the very significant sensitivity to the wider notion of observations (features) used by the classifiers, despite the violation of the probabilistic assumptions. For the easier NP task, the HMM model is competitive with the others when the classifiers used are NB or SNoW. In particular, a significant improvement in both probabilistic methods is achieved when their input is given by SNoW.

Our two main methods, PMM and CSCL, perform very well on predicting NP and SV phrases with CSCL at least as good as any other methods tried on these tasks. Both for NPs and

[4] Another solution in which the classifiers' suggestions inside each phrase are also accounted for is possible.

Table 1: Results ($F_{\beta=1}$) of different methods and comparison to previous works on NP and SV recognition. Notice that, in case of *simple*, the data with lexical features are too sparse to directly estimate the observation probability so we leave these entries empty.

	Method: Model	Method: Classifier	POS only	POS +words
NP	HMM	SNoW	90.64	92.89
		NB	90.50	92.26
		Simple	87.83	
	PMM	SNoW	90.61	92.98
		NB	90.22	91.98
		Simple	61.44	
	CSCL	SNoW	90.87	92.88
		NB	90.49	91.95
		Simple	54.42	
	Ramshaw & Marcus		90.6	92.0
	Argamon *et al.*		91.6	N/A
	Muñoz *et al.*		90.6	92.8
	Tjong Kim Sang & Veenstra		N/A	92.37
SV	HMM	SNoW	64.15	77.54
		NB	75.40	78.43
		Simple	64.85	
	PMM	SNoW	74.98	86.07
		NB	74.80	84.80
		Simple	40.18	
	CSCL	SNoW	85.36	90.09
		NB	80.63	88.28
		Simple	59.27	
	Argamon *et al.*		86.5	N/A
	Muñoz *et al.*		88.1	92.0

SVs, CSCL performs better than the probabilistic method, more significantly on the harder, SV, task. We attribute it to CSCL's ability to cope better with the length of the phrase and the long term dependencies.

Our methods compare favorably with others with the exception to SV in (Muñoz et al., 1999). Their method is fundamentally similar to our CSCL; however, they incorporated the features from open in the close classifier allowing to exploit the dependencies between two classifiers. We believe that this is the main factor of the significant difference in performance.

6 Conclusion

We have addressed the problem of combining the outcomes of several different classifiers in a way that provides a coherent inference that satisfies some constraints, While the probabilistic approach extends standard and commonly used techniques for sequential decisions, it seems that the constraint satisfaction formalisms can support complex constraints and dependencies more flexibly. Future work will concentrate on these formalisms.

References

S. Argamon, I. Dagan, and Y. Krymolowski. 1999. A memory-based approach to learning shallow natural language patterns. *Journal of Experimental and Theoretical Artificial Intelligence*, 10:1–22.

C. Cardie and D. Pierce. 1998. Error-driven pruning of treebanks grammars for base noun phrase identification. In *Proc. of ACL-98*, pages 218–224.

A. Carleson, C. Cumby, J. Rosen, and D. Roth. 1999. The SNoW learning architecture. Tech. Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May.

K. W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of ACL Conference on Applied Natural Language Processing*.

A. K. Mackworth. 1992. Constraint Satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. Vol. 1, 2^{nd} ed.

M. P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML-2000*.

N. Morgan and H. Bourlard. 1995. Continuous speech recognition. *IEEE Signal Processing Magazine*, 12(3):25–42.

M. Muñoz, V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In *Proc. of EMNLP-VLC'99*.

V. Punyakanok and D. Roth. 2000. Inference with classifiers. Tech. Report UIUCDCS-R-2000-2181, UIUC Computer Science Department, July.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of WVLC'95*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI'98*, pages 806–813.

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proc. of EACL'99*.