A Statistical Decision Making Method: A Case Study on Prepositional Phrase Attachment*

Mehmet Kayaalp, Ted Pedersen and Rebecca Bruce
Department of Computer Science & Engineering
Southern Methodist University
Dallas, TX 75275-0122
{kayaalp,pedersen,rbruce}@seas.smu.edu

Abstract

Statistical classification methods usually rely on a single best model to make accurate predictions. Such a model aims to maximize accuracy by balancing precision and recall. The Model Switching method as presented in this paper performs with higher predictive accuracy and 100% recall by using a set of decomposable models instead of a single one.

The implemented system, MS1, is tested on a case study, predicting Prepositional Phrase Attachment (PPA). The results show that it is more accurate than other statistical techniques that select single models for classification and competitive with other successful NLP approaches in PPA disambiguation. The Model Switching method may be preferable to other methods because of its generality (i.e., wide range of applicability), and its competitive accuracy in prediction. It may also be used as an analytical tool to investigate the nature of the domain and the characteristics of the data with the help of generated models.

1 Introduction

Decision problems are classically defined as problems whose answers fall in either of two classes: Yes and No (Garey and Johnson, 1979). Optimization problems are another class of problems that maximize or minimize some value; however, they can be cast as decision problems as well (Cormen et al., 1990). Classification problems incorporate the characteristics of both: A classification problem is a decision problem, in which a decision is made (a class is selected) that maximizes a utility function (von Neumann and Morgenstern, 1953). The Model Switching method as proposed in this paper can be used with any utility function (decision criterion) for any decision problem with categorical data that can be represented as a tuple $(C, F_1, F_2, ..., F_n)$ of a class variable C and some feature variables $F_{\{1...n\}}$.

In the following sections, we will describe the Prepositional Phrase Attachment (PPA) problem and various approaches to solving it. After discussing the statistical concepts used in this work, we will introduce the concept of Model Switching, why it is needed, how it works, and our experience on the PPA problem with Model Switching. Comparisons with earlier works on corpus-based PPA prediction and conclusions will follow.

2 PPA Problem

Resolving the PPA problem is a common problem in any NLP system that deals with syntactic parsing or text understanding. The Naive Bayes classifier and leading machine learning systems, such as C4.5 (Quinlan, 1993), CN2 (Clark and Niblett, 1989) and PEBLS (Cost and Salzberg, 1993), fail to provide prediction with competitive accuracy rates on this problem (see Table 4 on page 40). A sentence can be so ambiguous that it may not be possible to determine the correct attachment without extra contextual information. (Ratnaparkhi et al., 1994) reported that human experts could reach an accuracy of 93%, if cases were given as whole sentences out of context.

The PPA problem is illustrated by the following example:

I described the problem on the paper. (1)

This is an ambiguous sentence, which can be interpreted two different ways, depending on the site of PPA. The prepositional phrase (PP) in the above sentence is "on the paper." If it is attached to

*This research was supported in part by the Office of Naval Research under grant number N00014-95-1-0776

Mehmet Kayaalp, Ted Pedersen and Rebecca Bruce (1997) **A Statistical Decision Making Method: A Case Study on Prepositional Phrase Attachment.** In T.M. Ellison (ed.) *CoNLL97: Computational Natural Language Learning,* ACL pp 33-42.

the (object) noun "problem," then the interpretation would be equal to (2); on the other hand, if it is attached to the verb "describe," then it would be interpreted as (3).

$$\textit{I described the problem that was on the paper} \quad (2)$$

$$\textit{On the paper, I described the problem.} \quad (3)$$

In this paper, we address only the type of PPA problem illustrated above and don't consider other less frequent PPA problems. For the linguistic details of the problem, the reader can refer to (Hirst, 1987).

We use the PPA data created by (Brill and Resnik, 1994) and (Ratnaparkhi et al., 1994) to objectively compare the performances of the systems. Both data were extracted from the Penn Treebank Wall Street Journal (WSJ) Corpus (Marcus et al., 1993). In order to distinguish these data from each other, we call the former one B&R data and the latter one IBM data. Both PPA data were formatted in tuples with five variables (4), which denote the class (i.e., the PPA attachment site) and the features (i.e., verb, object noun, preposition and PP noun) in the respective order. Values of these variables for the above example (1) are illustrated in (5), where

$$(A, B, C, D, E) \quad (4)$$

$$(\text{verb}|\text{noun}, \text{"describe"}, \text{"problem"}, \text{"on"}, \text{"paper"}) \quad (5)$$

For representation convenience, we can map the values of these variables to positive integers as in Table 1. Then, the examples, (2) and (3) can be converted to tuples (6) and (7), respectively.

Levels	A	B	C	D	E
1	noun	describe	problem	on	paper
2	verb	join	board	as	director
3		be	dean	of	N.V.
⋮		⋮	⋮	⋮	⋮
81		improve	Hatch	plus	end
82		shipping	success		they
⋮		⋮	⋮		⋮
3845		develops	chunks		shots
3846			Koch		bar
⋮			⋮		⋮
5162			option		pot
5163					Citicorp
⋮					⋮
6625					rebate

Table 1: Substitution of variable values for associated integer labels at the Levels column. The number of levels of five variables are 2, 3845, 5162, 81 and 6625.

$$(A=1, B=1, C=1, D=1, E=1) \quad (6)$$

$$(A=2, B=1, C=1, D=1, E=1) \quad (7)$$

Using this convention, the PPA data can be represented in a contingency table (Table 2) with five dimensions, where each dimension is dedicated to a variable. The size of a contingency table is determined by the cardinality of values (a.k.a. levels) of these variables (8); for the IBM data, there are 2.13×10^{13} cells in the table (9). Each cell in the table corresponds to a unique combination of the variable values and all combinations are represented in the table.

		C	1					2	⋯	5162	
		B	1		2		⋯	1	⋯	3845	
		A	1	2	1	2	⋯	1	⋯	1	2
E	D										
1	1		**2**	**1**	0	0	⋯	0	⋯	0	0
	2		0	0	0	0	⋯	0	⋯	0	0
	⋮		⋮	⋮	⋮	⋮	⋯	⋮	⋯	⋮	⋮
2	1		0	0	0	0	⋯	0	⋯	0	0
	2		0	0	0	0	⋯	0	⋯	0	0
⋮	⋮		⋮	⋮	⋮	⋮	⋯	⋮	⋯	⋮	⋮
6625	81		0	0	0	0	⋯	0	⋯	**1**	0

Table 2: The PPA data can be represented in a $2 \times 3845 \times 5162 \times 81 \times 6625$ contingency table, where each cell contains frequency with which the corresponding 5-tuple (i.e., a unique PPA instance) occurs in the data.

$$(|A|=2, |B|=3845, |C|=5162, |D|=81, |E|=6625) \quad (8)$$

$$2 \times 3845 \times 5162 \times 81 \times 6625 = 2.13 \times 10^{13} \quad (9)$$

Considering that there are 27,937 PPA observations in the training and test data together, a search space of more than 21 trillion possible distinct cases (represented in the cells of contingency table) indicates that the data is extremely sparse.

To solve PPA problem, NLP researchers designed domain specific classifier systems. Those systems can be categorized in two classes:

1. Rule based systems (Boggess et al., 1991), (Brill and Resnik, 1994)

2. Statistical and information theoretic approaches (Hindle and Rooth, 1993), (Ratnaparkhi et al., 1994),(Collins and Brooks, 1995), (Franz, 1996)

Using lexical collocations to determine PPA with statistical techniques was first proposed by (Hindle and Rooth, 1993). They suggested a score called Lexical Association to predict PPA. It is a log likelihood ratio of probability estimates of two PPA sites. The probability of attachment was based on the frequencies of the 2-tuples (B, D), and (C, D), where B, C, D stand for the variables: verb, object noun and preposition, respectively. While (Hindle and Rooth, 1993) stated that this approach was not successful in estimating PPA using small 2-tuple frequencies, which comprised a major portion of the PPA data, the accuracy reported was 79.7%, which is a substantial improvement over the lower bound of 65% (10):

$$\max\left\{\frac{f(A=1)}{f(A=1)+f(A=2)}, \frac{f(A=2)}{f(A=1)+f(A=2)}\right\} \tag{10}$$

The lower bound for the B&R data is 63% (Brill and Resnik, 1994) and for the IBM data is 52% (Ratnaparkhi et al., 1994).

(Ratnaparkhi et al., 1994) was the first to considered the full four feature set defined in (4). The approach made use of a maximum entropy model (Berger et al., 1996) formulated from frequency information for various combinations of the observed features. The combinations that reduced the entropy most, were chosen. The accuracy of PPA classification using this approach was 77.7% on the IBM data. (For performance comparison of various approaches on available data, please refer to Table 4 on page 40.)

(Brill and Resnik, 1994) suggested a rule based approach where the antecedent of each rule specifies values for the feature variables in (4). A typical rule might be as follows:

$$features(B=12, C, D=3, E) \rightarrow ppa(A=1) \tag{11}$$

471 such inference rules are found useful and ordered to reduce the error-rate to a minimum. They reported an accuracy of 80.8% on the data that we also use. They also duplicated the experiment of (Hindle and Rooth, 1993), which scored around 5% less than the rule-based approach.

(Collins and Brooks, 1995) proposed a specific heuristic computation to predict PPAs. The idea originated from the back-off model (Katz, 1987). If the combination of feature values observed for a test instance is also observed in the training set, then that test instance is classified with the most frequent PPA site for those feature values in the training set. Otherwise, probability estimates for the two PPA sites are obtained from functions(12)–(14), via a process similar to model switching. If the highest complexity formulation, (12), cannot be used to classify a test instance (i.e., the required feature value combinations are not observed in the training data), then the decision process is *switched* to the next function, where functions are ranked based on complexity (i.e., the arity of the frequency distributions used in the function). If all fail, the assignment is noun attachment, since 52% of the time the attachment site on the training data was noun.

$$\hat{p}(A|B,C,D,E) = \frac{f(A,B,C,D)+f(A,B,D,E)+f(A,C,D,E)}{f(B,C,D)+f(B,D,E)+f(C,D,E)} \tag{12}$$

$$\hat{p}(A|B,C,D,E) = \frac{f(A,B,D)+f(A,C,D)+f(A,D,E)}{f(B,D)+f(C,D)+f(D,E)} \tag{13}$$

$$\hat{p}(A|D) = \frac{f(A,D)}{f(D)} \tag{14}$$

If a higher order function cannot classify a test instance, then the decision process is *switched* to the next function. If all fail, the guess is the noun attachment, since 52% of the time the attachment site on the training data was noun.

While the probability estimates in (14) are maximum likelihood estimates (MLEs), the estimates in (12) and (13) are heuristic formulations (i.e., not MLEs). The rationale behind these formulae are:

1. a decision made by utilizing more feature variables should be favorable over the others,
2. the preposition feature D is essential; thus, it is better to keep it in all n-grams of the decision functions.

They used IBM data, which we also use, and reported an accuracy of 84.1%.

(Franz, 1996) proposed a new feature set, which provided a more compact representation of the PPA data. Using a hierarchical log-linear model containing only second order interactions, he achieved a classification performance comparable to that of (Hindle and Rooth, 1993). He also designed another experiment with a less common PPA problem with three attachment sites.

3 Decomposable Models

In this paper, PPA is cast as a problem in supervised learning, where a probabilistic classifier is induced from tagged training data in the form of 5-tuples (6) and (7). The task is to predict the value of the tag A given the values of the feature variables B through E.

Probabilistic models (e.g., decomposable models) specify joint distribution functions that assign probability values to every unique combination of the model variables, where the sum of those values is equal to 1. We adopt a Maximum Likelihood Estimation (MLE) approach. Given a decomposable model, MLE yields the most probable tag to each test data instance represented by a 4-tuple of feature values.

Decomposable models belong to the class of *graphical models*,[1] where variables are either interdependent or conditionally independent of one another.[2] All graphical models have a graphical representation such that each variable in the model is mapped to a vertex in the graph, and there is an undirected edge between each pair of vertices corresponding to a pair of interdependent variables. While edges represent interactions between pairs of variables, i.e., second order interactions, cliques[3] with n vertices represent n^{th} order interactions. Any two vertices that are not directly connected by an edge are conditionally independent given the values of the vertices on the path that connects them.

Decomposable models are graphical models that are isomorphic to chordal graphs. In chordal graphs, there is no cycle of four or more without a chord, where a cord is an edge joining two non-consecutive vertices on the cycle. The elementary components of a chordal graph are its cliques; therefore, a chordal graph can be represented as a set of its cliques.

The chordal graph in Figure 1 represents a decomposable model, which we can mnemonically denote as (15).

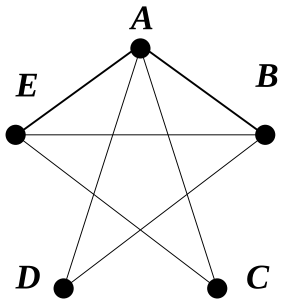

Figure 1: The decomposable model $ABD.ABE.ACE$. Edges of the separators, AB and AE (corresponding to $ABD \cap ABE$ and $ABE \cap ACE$), are drawn thicker. A separator is a set of vertices whose removal disconnects the graph.

$$ABD.ABE.ACE \qquad (15)$$

In this model, variables A, B, and D are stochastically dependent since they form a clique. Similar statements can be made for the other cliques in the model. The interactions between AB and AE, denoted by the corresponding edges $\overline{AB}$, $\overline{AE}$ are observed in two out of the three cliques which indicates their relative importance in describing this distribution. The variable A is observed in all three cliques of the model because we consider only those cliques that contain the class variable A in defining the model. There are three edges missing, $\overline{BC}$, $\overline{CD}$, and $\overline{DE}$, which distinguish this model from the saturated model $ABCDE$. These missing edges denote three conditional independence relations:

1. The variables D and E are conditionally independent given AB (intersection of two cliques, $ABD \cap ABE$).
2. The variables B and C are conditionally independent given AE ($ABE \cap ACE$).
3. The variables C and D are conditionally independent given A ($ABD \cap ACE$).

This approach to classifying PPA is the first to make use of conditional independence in modeling the distribution of feature variables.

A well known example of a decomposable model is the *Naive Bayes* model in which all feature variables are conditionally independent given the value of classification variable. For the PPA problem, the Naive Bayes model is $AB.AC.AD.AE$.

Decomposable models are important because they are those graphical models that express the joint probability distributions of the variables in terms of the product of their marginal distributions, where each factor of the product corresponds to a clique or a separator in the graphical representation of the model. Because the joint distribution functions of decomposable models have such closed-form expressions, the parameters as Maximum Likelihood Estimates (MLEs) can be calculated directly from the training data without the need for an iterative fitting procedure; hence, those MLEs are also called direct estimates (Bishop et al., 1975).

3.1 Maximum Likelihood Estimation

Let the PPA variables, $|A| = I, |B| = J, \ldots, |E| = M$ resulting in an $I \times J \times K \times L \times M$ contingency table (e.g., Table 2). Let the count in each cell (i.e., the frequency with which the corresponding 5-tuple is observed in the training data) be denotes as n_{ijklm}. When all variables are considered to be interdependent (i.e., the saturated decomposable model) the maximum likelihood estimate of the probability of any 5-tuple is equal to the count in the corresponding cell n_{ijklm} divided by the total count N, which is equal to 24,840 for the IBM training data (Table 2).

$$\hat{p}(A=1, B=1, C=1, D=1, E=1) =$$

[1] Graphical models are a subset of log-linear models.

[2] B and C are conditionally independent given A if $P(B|C,A) = P(B|A)$.

[3] A clique is a complete (sub)graph, where every vertex pair is connected with an edge.

$$\hat{p}_{11111} = \frac{n_{11111}}{N} = \frac{2}{24840} \tag{16}$$

Estimates of the marginal probability distributions can be calculated in a similar fashion. If we are interested in the probability of observing a verb attachment when "describe" is the noun, and "on" is the preposition (i.e., $A = 1, B = 1, D = 1$), regardless of the values of the other variables, it can be calculated as in (17) and (18).

$$n_{11+1+} = \sum_{k=1}^{K}\sum_{m=1}^{M} n_{11k1m} \tag{17}$$

$$\hat{p}(A=1, B=1, D=1) = \hat{p}_{11+1+} = \frac{n_{11+1+}}{N} \tag{18}$$

Let c denote the specific cell coordinates (e.g., 11111 in (16)), and let the model $\mathcal{M} = \{\mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots \mathcal{C}_D\}$, where $\mathcal{C}_d$ denotes a clique in the graph representation of $\mathcal{M}$, then the direct estimates (MLEs) are computed as in (19).

$$\hat{m}(c) = \frac{\prod_{d=1}^{D} \hat{p}(c_{\mathcal{C}_d})}{\prod_{d=2}^{D} \hat{p}(c_{\mathcal{S}_d})} \tag{19}$$

where the factors in the numerator are the marginal probabilities for c in the cliques $\{\mathcal{C}_d\}$, whose union represents the model. The intersections of cliques $\{\mathcal{C}_d\}$ yield separators $\{\mathcal{S}_d\}$ and the marginal probabilities for c in $\{\mathcal{S}_d\}$ are factors in the denominator (Lauritzen, 1996). For the saturated model, $\{\mathcal{S}_d\} = \{\}$, and the MLE is most straightforward:

$$\begin{aligned} \hat{m}(c) &= \hat{p}_c & (20) \\ \hat{m}_{11111} &= \hat{p}_{11111} & (21) \end{aligned}$$

MLEs of the model (15) can be computed as in (22), and using this model, MLEs of the examples (2) and (3) can be calculated as in (23) and (24), respectively.

$$\begin{aligned} \hat{m}(c) &= \frac{\hat{p}(A,B,D)\,\hat{p}(A,B,E)\,\hat{p}(A,C,E)}{\hat{p}(A,B)\,\hat{p}(A,E)} & (22) \\ \hat{m}_{11111} &= \frac{\hat{p}_{11+1+}\,\hat{p}_{11++1}\,\hat{p}_{1+1+1}}{\hat{p}_{11+++}\,\hat{p}_{1+++1}} & (23) \\ \hat{m}_{21111} &= \frac{\hat{p}_{21+1+}\,\hat{p}_{21++1}\,\hat{p}_{2+1+1}}{\hat{p}_{21+++}\,\hat{p}_{2+++1}} & (24) \end{aligned}$$

As seen in this example, decomposable models provide us not only a very powerful representation medium but also computational efficiency in estimating parameters.

4 Model Switching

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be equal to MLEs in (23) and (24). There are four cases in determining the class based on these equations.

$$\begin{aligned} \mathcal{E}_1 = 0 \wedge \mathcal{E}_2 = 0 &\rightarrow A = \text{null} & (25) \\ \mathcal{E}_1 > 0 \wedge \mathcal{E}_1 = \mathcal{E}_2 &\rightarrow A = \text{null} & (26) \\ \mathcal{E}_1 > \mathcal{E}_2 &\rightarrow A = 1 & (27) \\ \mathcal{E}_1 < \mathcal{E}_2 &\rightarrow A = 2 & (28) \end{aligned}$$

In cases (25) and (26), there is no classification and no recall for this test instance with this model. In (27) and (28), the classifications are noun and verb attachments, respectively.

For the PPA data with five variables, there are only 110 decomposable models, corresponding to all chordal graphs of order five or less, where every clique of the order two and higher contains the vertex that represents the class variable. Since this number is not large, we considered all of these models for classification.[4] Let all test instances be composed in the set $\mathcal{T}$ and let

$$\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_m \tag{29}$$

where $\mathcal{T}_i$ is a set of test instances that *can* be classified with model $\mathcal{M}_i$ for $(1 \le i \le m = 110)$; i.e., the outcomes of $\hat{m}(A|\mathcal{T}_i, \mathcal{M}_i)$ is either in (27) or in (28).

These estimates may not always be correct, unless the information in features are sufficient and the classification model is perfect; therefore, each set of estimates associated with $\mathcal{T}_i$ and $\mathcal{M}_i$ has a precision value:

$$\begin{aligned} precision(\mathcal{M}_i|\mathcal{T}_i) &= \frac{|\mathcal{T}_{i\mathcal{C}}|}{|\mathcal{T}_i|} & (30) \\ \mathcal{T}_i &= \mathcal{T}_{i\mathcal{C}} \cup \mathcal{T}_{i\mathcal{W}} & (31) \end{aligned}$$

where $\mathcal{T}_{i\mathcal{C}}$ and $\mathcal{T}_{i\mathcal{W}}$ are sets of correctly and wrongly classified test instances in set $\mathcal{T}_i$. If we have an ordered list of models $(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m)$ as a certificate, where

$$precision(\mathcal{M}_i|\mathcal{T}_i) \ge precision(\mathcal{M}_{i+1}|\mathcal{T}_{i+1}) \tag{32}$$

we could use the certificate to maximize the overall classification accuracy.

Since the first model $\mathcal{M}_1$ is associated with the highest precision value, the probability that a test instance is correctly classified with $\mathcal{M}_1$ is higher than

[4] For problems with larger variable set additional techniques (Edwards and Havránek, 1987) or (Madigan and Raftery, 1994) are necessary to reduce the model space.

that probability for any other model; therefore, $\mathcal{M}_1$ should be used to classify all possible test instances.

$$\mathcal{T} = \mathcal{T}^0 = \mathcal{T}_1 \cup \mathcal{T}^1, \text{ where } \mathcal{T}_1 \cap \mathcal{T}^1 = \{\} \quad (33)$$

After $\mathcal{T}_1$ is classified, the process is repeated for the remaining test instances $\mathcal{T}^1$ with $\mathcal{M}_2$ that is the most "precise" model remaining in the model set. This cycle can be generalized as

$$\mathcal{T}^{i-1} = \mathcal{T}_i \cup \mathcal{T}^i, \text{ where } \mathcal{T}_i \cap \mathcal{T}^i = \{\}, \text{ and } \mathcal{T} = \mathcal{T}^0 \quad (34)$$

and will be iterated k times, where $\mathcal{T}^k = \{\}$. The overall classification accuracy then be calculated as

$$accuracy(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k|\mathcal{T}) = \frac{\sum_{i=1}^{k} |\mathcal{T}_{i\mathcal{C}}|}{|\mathcal{T}|} \quad (35)$$

The question remains now, how we can find the list of models $(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k, \ldots, \mathcal{M}_m)$ ordered by precision. Since precision is a measure that can be acquired after classifying all test instances, how can we order models based on precision before testing?

One approach is to use the error rates of the models acquired through cross-validation. The technique we use here is called leave-one-out cross-validation (Lachenbruch and Mickey, 1968). Let the training data set be $\mathcal{R}$, where every data instance $\rho_i \in \mathcal{R}$, $i = 1, 2, \ldots, r$ and $r = |\mathcal{R}|$. When a model $\mathcal{M}_j$ is applied to a data instance ρ_i, in this technique, all training instances except ρ_i (i.e., $\mathcal{R} - \rho_i$) are used to compute the direct estimate for ρ_i. This process is repeated for every data instance (i.e., r times). This technique is applied to all training instances for every model. The precision score of each model is collected, and based on those scores, the models are ordered.

If k (the number of models used to classify all PPA instances) is small, then it is expected that after each iteration the test instances remaining to be classified would be decreased significantly; hence, the characteristics of $\mathcal{T}^{i-1}$ and $\mathcal{T}^i$ might differ substantially and ordering the remaining models based on $\mathcal{T}^i$, rather than $\mathcal{T}^0$, might increase the overall accuracy.

A second experiment is designed to apply this recursive strategy to order the models via the same cross-validation process. First, the most precise model for the entire (training) data is identified. Then, the data instances that are classified with the first model are excluded from the original data set, as in (33). Within the remaining data instances, all models in $\{\mathcal{M}_2, \mathcal{M}_3, \ldots, \mathcal{M}_m\}$ are searched for the current most precise model. This model selection cycle is iterated exhaustively (34) until all data instances are classified. The models selected for the IBM data are shown in Table 3.

Models	Cor	Inc	Prc	Acc	RTC
$ABCDE$	150	17	90	89.8	2930
$ABDE.ACD$	145	16	90	89.9	2769
$ACDE.ABD$	192	10	95	91.9	2567
$ABDE$	46	11	81	90.8	2510
$ACDE.ABE$	5	0	100	90.9	2505
$ABCD$	293	42	87	89.6	2170
$ABD.ACD.ADE$	441	73	86	88.3	1656
$ACDE$	51	11	82	88.0	1594
$ABD.ACD$	263	50	84	87.3	1281
$ABE.ACD.ACE$	3	0	100	87.4	1278
ABD	401	107	79	85.5	770
ACD	296	63	82	85.1	411
$ABE.ACE.ADE$	0	0	0	85.1	411
$ACE.ADE.AB$	6	1	86	85.1	404
ADE	156	47	77	84.5	201
AD	141	56	72	83.7	4
$AD.AE$	0	0	0	83.7	4
$ABC.AE$	1	1	50	83.7	2
$AC.AD$	0	0	0	83.7	2
A	2	0	100	83.7	0

Table 3: Classification with Multiple Models. Cor (Inc): Number of correct (incorrect) classifications. Prc: Precision ×100. Acc: Accuracy ×100. RTC: Remaining Test Cases.

The MLE algorithm is a table look up, where each table contains marginal values for a clique of variables as defined in the graph representation. If those values could be stored in a memory array, the time complexity of MLE could be $\Theta(1)$; however, the number of values is huge, thus we have to store each set of clique marginals on disk, and currently the access to the data is through sequential file access with a time complexity $\Theta(n)$, where n is the number of training instances. MLEs need to be computed for m models and for n training instances. During each recursive step a considerable part of the training instances are classified (around 5%); thus we may represent the process as

$$\begin{aligned} N &= mn^2 && (36) \\ T(N) &= T(\tfrac{19}{20}N) + N && (37) \\ &\Longrightarrow \Theta(N \log N) && (38) \end{aligned}$$

Therefore, the average time complexity for the current program is $\Theta(mn^2 \log(mn^2))$, but through memoization,[5] the overhead of the recursion will be drastically reduced in newer versions of the program.

[5] A standard dynamic programming technique that stores computed information in a table, which is looked up when that information is needed next time.

The software of MS1 is developed in Perl and is freely available for research purposes only. Interested parties may contact the first author.

5 Discussion

In some of the earlier works on PPA there are aspects of the model switching framework. For example, (Brill and Resnik, 1994) ordered rules to minimize the error-rate in PPA classification. Each of these inference rules may be considered a decision function in a decision list. Whenever a higher order rule fails, the control *switches* to the next rule to classify that test instance. (Collins and Brooks, 1995) ordered heuristic decision functions by complexity (arity) and classified test instances with the most complex applicable function.

Non-recursive Model Switching consists of two phases:

1. Ordering available models (e.g., via leave-one-out cross-validation),
2. Applying the model on top of the list to the test data; whenever that model does not yield any estimate, the system switches to the next model on the list.

The first phase corresponds to the learning phase of learning systems; whereas, the last phase can be conceptualized as a decision list (Rivest, 1987) and (Kohavi and Benson, 1993), where the control is conditioned by the availability of a direct estimate given a model with a test instance.[6]

In the recursive version of the Model Switching, however, the model list is dynamically changed since the above phases are within a loop, where in each iteration all instances of the available data are considered for classification and those which are classified are excluded from the data for the next iteration. The base case of recursion is reached when all instances are classified.

Although in this work we suggest a precision-driven model ordering scheme, the Model Switching method enables one to use any other utility function such as accuracy or F-measure. There are other utility functions that need not be acquired through cross-validation, but rather can be collected by analyzing the entire training set as in statistical significance analysis (e.g., G^2, Pearson's χ^2), or information criteria (e.g., Akaike or Bayes Information Criteria etc.), which can be used as well.

An advantage of this method is that we make use of a complex and powerful set of models. Much of the earlier PPA research was confined to single clique models, such as $ABCD$ or AB, which are a small subset of decomposable models.

5.1 Quantitative Analysis

Statistical (decomposable) model selection techniques were first applied to NLP problems by (Bruce and Wiebe, 1994). Those model selection techniques aim to find a single best model but they alone do not perform as well as Model Switching, since even the most accurate decomposable model, $AB.AD$, had a classification accuracy of 77%.

Unlike Model Switching, the methods suggested in earlier PPA works are usually tailored to the PPA problem, thus it is hard to transfer them to another domain. On the other hand, neither Naive Bayes nor the conventional machine learning tools, such as CN2, C4.5 and PEBLS, perform as well. These four symbolic classifiers are well known and are diverse to some extent: Naive Bayes is a simple Bayesian approach, CN2 is based on rule induction, C4.5 is based on decision trees, and PEBLS is based on nearest neighbor method. A performance comparison of various classifiers with MS1 is given on Table 4. The comparison between the proposed systems solving PPA ambiguity and general machine learning systems was always neglected in earlier articles on PPA problem.[7]

The results of the first five classifiers presented in Table 4 and the performance of B&R classifier on IBM data were determined as part of this study, while the other four results are benchmarks quoted from the authors cited above. Those benchmarks were produced via single trials, hence we performed single trial tests as well. CN2, C4.5 and PEBLS performances were based on their default settings. The only exception involved CN2 where an ordered-induced-rule-list is used instead of an unordered one, since the ordered rules yield 99.7% accuracy versus 90.8% accuracy of unordered rules on the IBM training data. After the test, we checked the accuracy rates of unordered induced rules, which are unexpectedly better than the ordered ones: 78% on B&R data and 76.2% on IBM data. Naive Bayes' recall values are very low: 74% for IBM data and 78% for B&R data; therefore, the remaining test instances are classified as the most frequent class. Notice that this is also a type of model switching, where the forms of the models and the model list $\mathcal{M} = (AB.AC.AD.AE, A)$ are predetermined as done by (Collins and Brooks, 1995).

[6] This relevance of decision lists was indicated by Mike Collins in our personal discussions.

[7] (Ratnaparkhi et al., 1994) reported a decision tree experiment using mutual information with 77.7% accuracy.

Data	Classifiers							
	Bayes	CN2	C4.5	PEBLS	**MS1**	B&R	IBM	C&B
B&R	74.6	77.4	78.4	76.4	**81.2**	80.8	n.a.	81.9
IBM	73.0	70.7	79.6	76.9	**83.7**	81.4	77.7	84.1

Table 4: Performances of various classifiers on available data. C&B:(Collins and Brooks, 1995); B&R: (data/classifier) by (Brill and Resnik, 1994); IBM: (data/classifier) by (Ratnaparkhi et al., 1994); Bayes: Naive Bayes with defaults, i.e., $\mathcal{M} = (AB.AC.AD.AE, A)$.

The performance differences between MS1 and C&B, the Back-off Model by (Collins and Brooks, 1995), are 0.4% for IBM data and 0.7% for B&R data. With only two test trials and without any deviation measure these differences cannot be considered significant, especially in this case, where the performances of the classifiers fluctuate 2–3% (e.g., C&B accuracy deviates 2.2%) within two very similar data sets, B&R and IBM data. As one anonymous reviewer indicated, the 0.7% accuracy difference on B&R data needs to be evaluated cautiously due to the size of the B&R test data, which contains only 500 test instances; whereas the IBM data contains 3097 test instances.

5.2 Qualitative Analysis

The approach of (Collins and Brooks, 1995) is simpler than MS1, since it doesn't consist of any learning part; the models were selected and grouped by its designers and ordered heuristically, which means classification requires *prior* knowledge specific to the domain. With the human expertise involved, the list of models is simpler and shorter than the list found by MS1 and it is heuristically grouped and weighted (forming a kind of mixture model), which is not the case in MS1 at this point in time; nevertheless, MS1 reached to a performance level that is competitive to the other system supported with human expertise. MS1 uses neither any lexical information nor heuristics with respect to the PPA problem; hence, it can be adopted and applied to any other classification problem involving categorical data. MS1 is a machine learning alternative to the system developed by (Collins and Brooks, 1995), and the ordering of the models that it produces may provide insight into the data that could aid in developing a custom mixture model.

Unlike the other techniques, MS1 generates an ordered list of models where each model provides a graphical representation of the interdependencies among variables. The user can identify relevant relations and see which features play the most significant roles; thus, one can not only predict the outcome of a classification problem with high accuracy but also gain insight into the nature of the domain and the data under investigation. For example, MS1 identified the fact that the preposition feature (variable D) is so important that all test instances (except the last four) were predicted by models that have this variable. This was one of the most important heuristic steps in formulating the approach used by (Collins and Brooks, 1995). Further analysis of the model list by linguists may yield other observations, such as, in the first 75% of the predictions, 97% of the test instances were identified using models containing the interaction ABD with a precision of 86%, and in the rest of the predictions this interaction was not useful. Similar model lists can be generated on various corpora and their comparisons may reveal differences in those corpora.

MS1 and the systems by (Ratnaparkhi et al., 1994) and (Brill and Resnik, 1994) consist of a training phase, where they form certain structures (such as rules, models, etc.) that are used with the available statistics to classify test instances; therefore, these systems can be considered true learning systems. On the other hand in systems designed by (Hindle and Rooth, 1993), (Collins and Brooks, 1995), and (Franz, 1996), the forms of models were predetermined by their designers, as in the Naive Bayes approach.

5.3 Scalability

The structure of the underlying PPA data (4) casts a difficult problem to learning system. When the number of observations grows, the levels of features (except that of the preposition, which is limited by grammar) grow proportionally. This effect was first identified by (Zipf, 1935). Due to this effect the number of cells in contingency table representations explodes, which corresponds to an exponential growth in the search space.

Three general machine learning systems cited above require very large main memory capacity to run the PPA data, which brings the scalability into question. MS1's implementation is based on large data and limited main memory assumptions, hence computation time has been traded with memory requirement. The Model Switching approach is scalable in computation time and memory: While the data size grows, the leave-one-out cross-validation technique may be switched to a simpler *v-fold cross-validation* technique, which is "stable" and preferable for larger data size (Breiman et al., 1984). There is always, a much simpler choice: Ranking models through statistical significance analysis or through information criteria, whose cost is $\Theta(|\mathcal{M}|)$.

One problem encountered in applying Model Switching to other domains is that the number of decomposable models grows exponentially with the number of possible variables. The method of (Edwards and Havránek, 1987) or (Madigan and Raftery, 1994) for selecting a good subset of models for the data resolves this last concern regarding scalability. Using these techniques, the Model Switching method may be applied to other NLP problems with much larger size of feature variables. Model Switching method is currently being applied to word sense disambiguation which is cast with eight features. The preliminary results are very encouraging, and provide evidence for the robustness of the methodology.

6 Acknowledgments

We gratefully acknowledge the support provided for this research by the Office of Naval Research under grant number N00014-95-1-0776. We would also like to thank Mike Collins for his constructive comments.

References

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–68.

Yvonee M. M. Bishop, Stephen E. Fienberg, and Paul W. Holland. 1975. *Discrete Multivariate Analysis: Theory and Practice*. The MIT Press, Cambridge, MA.

Lois Boggess, Rajeev Agarwal, and Ron Davis. 1991. Disambiguation of prepositional phrases in automatically labeled technical text. In *Proceeding of the Ninth National Conference on Artificial Intelligence*, pages 155–159, Cambridge, MA. AAAI, MIT Press.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING-94)*.

Rebecca Bruce and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*.

Peter Clark and Tim Niblett. 1989. The CN2 induction algorithm. *Machine Learning*, 3:261–283.

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Scott Cost and Steven Salzberg. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

David Edwards and Thomáš Havránek. 1987. A fast model selection procedure for large families of models. *Journal of American Statistical Association*, 82(397):205–213.

Alexander Franz. 1996. Learning PP attachment from corpus statistics. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Artificial Intelligence*, pages 188–202. Springer-Verlag, New York, NY.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability*. W. H. Freeman and Company, New York, NY.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Graeme Hirst. 1987. *Semantic interpretation and the resolution of ambiguity*. Cambridge University Press, New York, NY.

Slava M. Katz. 1987. Estimation of probabilities from data for the language model component of a speech recognizer. In *Transactions* *on Acoustics, Speech, and Signal Processing*, pages 400–401. IEEE.

Ron Kohavi and Scott Benson. 1993. Research on decision lists. *Machine Learning*, 13:131–134.

Peter A. Lachenbruch and M. Ray Mickey. 1968. Estimation of error rates in discriminant analysis. *Technometrics*, 10(1):1–11, February.

Steffen L. Lauritzen. 1996. *Graphical Models*. Oxford University Press, New York, NY.

David Madigan and Adrian E. Raftery. 1994. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of American Statistical Association*, 89(428):1535–1546.

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

John Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers, San Mateo, CA.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of Human Language Technology Workshop*, pages 250–255, Plainsboro, NJ. ARPA.

Ronald L. Rivest. 1987. Learning decision lists. *Machine Learning*, 2:229–246.

John von Neumann and Oskar Morgenstern. 1953. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

George Kingsley Zipf. 1935. *The Psycho-biology of Language*. Houghton Mifflin Company, Boston, MA.