Competitive generative models with structure learning for NLP classification tasks

Kristina Toutanova
Microsoft Research
Redmond, WA
kristout@microsoft.com

Abstract

In this paper we show that generative models are competitive with and sometimes superior to discriminative models, when both kinds of models are allowed to learn structures that are optimal for discrimination. In particular, we compare Bayesian Networks and Conditional log-linear models on two NLP tasks. We observe that when the structure of the generative model encodes very strong independence assumptions (*a la* Naive Bayes), a discriminative model is superior, but when the generative model is allowed to weaken these independence assumptions via learning a more complex structure, it can achieve very similar or better performance than a corresponding discriminative model. In addition, as structure learning for generative models is far more efficient, they may be preferable for some tasks.

1 Introduction

Discriminative models have become the models of choice for NLP tasks, because of their ability to easily incorporate non-independent features and to more directly optimize classification accuracy. State of the art models for many NLP tasks are either fully discriminative or trained using discriminative reranking (Collins, 2000). These include models for part-of-speech tagging (Toutanova et al., 2003), semantic-role labeling (Punyakanok et al., 2005; Pradhan et al., 2005b) and Penn Treebank parsing (Charniak and Johnson, 2005).

The superiority of discriminative models has been shown on many tasks when the discriminative and generative models use exactly the same model structure (Klein and Manning, 2002). However, the advantage of the discriminative models can be very slight (Johnson, 2001) and for small training set sizes generative models can be better because they need fewer training samples to converge to the optimal parameter setting (Ng and Jordan, 2002). Additionally, many discriminative models use a generative model as a base model and add discriminative features with reranking (Collins, 2000; Charniak and Johnson, 2005; Roark et al., 2004), or train discriminatively a small set of weights for features which are generatively estimated probabilities (Raina et al., 2004; Och and Ney, 2002). Therefore it is important to study generative models and to find ways of making them better even when they are used only as components of discriminative models.

Generative models may often perform poorly due to making strong independence assumptions about the joint distribution of features and classes. To avoid this problem, generative models for NLP tasks have often been manually designed to achieve an appropriate representation of the joint distribution, such as in the parsing models of (Collins, 1997; Charniak, 2000). This shows that when the generative models have a good model structure, they can perform quite well.

In this paper, we look differently at comparing generative and discriminative models. We ask the question: given the same set of input features, what is the best a generative model can do if it is allowed to learn an optimal *structure* for the joint distribution, and what is the best a discriminative model can do if it is also allowed to learn an optimal structure. That is, we do not impose any independence assumptions on the generative or discriminative models and let them learn the best representation of the data they can.

Structure learning is very efficient for generative models in the form of directed graphical models (Bayesian Networks (Pearl, 1988)), since the optimal parameters for such models can be estimated in closed form. We compare Bayesian Networks with structure learning to their closely related discriminative counterpart – conditional log-linear models with structure learning. Our conditional log-linear models can also be seen as Conditional Random Fields (Lafferty et al., 2001), except we do not have a structure on the labels, but want to learn a structure on the features.

We compare the two kinds of models on two NLP classification tasks – prepositional phrase attachment and semantic role labelling. Our results show that the generative models are competitive with or better than the discriminative models. When a small set of interpolation parameters for the conditional probability tables are fit discriminatively, the resulting hybrid generative-discriminative models perform better than the generative only models and sometimes better than the discriminative models.

In Section 2, we describe in detail the form of the generative and discriminative models we study and our structure search methodology. In Section 3 we present the results of our empirical study.

2 Model Classes and Methodology

2.1 Generative Models

In classification tasks, given a training set of instances $D = \{[x_i, y_i]\}$, where x_i are the input features for the i-th instance, and y_i is its label, the task is to learn a classifier that predicts the labels of new examples. If $\mathcal{X}$ is the space of inputs and $\mathcal{Y}$ is the space of labels, a classifier is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$. A generative model is one that models the joint probability of inputs and labels $P_D(x, y)$ through a distribution $P_\theta(x, y)$, dependent on some parameter vector θ. The classifier based on this generative model chooses the most likely label given an input according to the conditionalized estimated joint distribution. The parameters θ of the fitted distribution are usually estimated using the maximum joint likelihood estimate, possibly with a prior.

We study generative models represented as Bayesian Networks (Pearl, 1988), because their parameters can be estimated extremely fast as the maximizer of the joint likelihood is the closed form relative frequency estimate. A Bayesian Network is an acyclic directed graph over a set of nodes. For every variable Z, let $Pa(Z)$ denote the set of parents of Z. The structure of the Bayesian Network encodes the following set of independence assumptions: every variable is conditionally independent of its non-descendants given its parents. For example, the structure of the Bayesian Network model in Figure 1 encodes the independence assumption that the input features are conditionally independent given the class label.

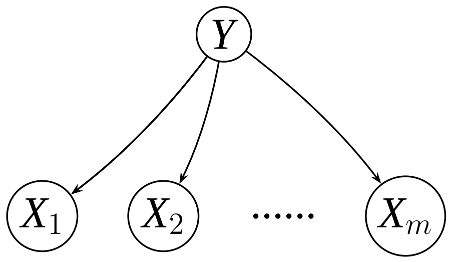


Figure 1: Naive Bayes Bayesian Network

Let the input be represented as a vector of m nominal features. We define Bayesian Networks over the m input variables $X_1, X_2, \ldots, X_m$ and the class variable Y. In all networks, we add links from the class variable Y to all input features. In this way we have generative models which estimate class-specific distributions over features $P(X|Y)$ and a prior over labels $P(Y)$. Figure 1 shows a simple Bayesian Network of this form, which is the well-known Naive Bayes model.

A specific joint distribution for a given Bayesian Network (BN) is given by a set of conditional probability tables (CPTs) which specify the distribution over each variable given its parents $P(Z|Pa(Z))$. The joint distribution $P(Z_1, Z_2, \ldots, Z_m)$ is given by:

$$P(Z_1, Z_2, \ldots, Z_m) = \prod_{i=1 \ldots m} P(Z_i|Pa(Z_i))$$

The parameters of a Bayesian Network model given its graph structure are the values of the conditional probabilities $P(Z_i|Pa(Z_i))$. If the model is trained through maximizing the joint likelihood of the data, the optimal parameters are the relative frequency estimates: $\hat{P}(Z_i = v|Pa(Z_i) = \vec{u}) = \frac{count(Z_i=v, Pa(Z_i)=\vec{u})}{count(Pa(Z_i)=\vec{u})}$ Here v denotes a value of Z_i and $\vec{u}$ denotes a vector of values for the parents of Z_i.

Most often smoothing is applied to avoid zero probability estimates. A simple form of smoothing is add-α smoothing which is equivalent to a Dirichlet prior. For NLP tasks it has been shown that other smoothing methods are far superior to add-α smoothing – see, for example, Goodman

(2001). In particular, it is important to incorporate lower-order information based on subsets of the conditioning information. Therefore we assume a structural form of the conditional probability tables which implements a more sophisticated type of smoothing – interpolated Witten-Bell (Witten and Bell, 1991). This kind of smoothing has also been used in the generative parser of (Collins, 1997) and has been shown to have a relatively good performance for language modeling (Goodman, 2001).

To describe the form of the conditional probability tables, we introduce some notation. Let Z denote a variable in the BN and $Z_1, Z_2, \ldots, Z_k$ denote the set of its parents. The probability $P(Z = z|Z_1 = z_1, Z_2 = z_2, \ldots, Z_k = z_k)$ is estimated using Witten-Bell smoothing as follows: (below the tuple of values $z_1, z_2, \ldots, z_k$ is denoted by z_{1k}).

$$P_{WB}(z|z_{1k}) = \lambda(z_{1k}) \times \hat{P}(z|z_{1k}) + (1 - \lambda(z_{1k})) \times P_{WB}(z|z_{1k-1})$$

In the above equation, $\hat{P}$ is the relative frequency estimator. The recursion is ended by interpolating with a uniform distribution $\frac{1}{V_z}$, where V_z is the vocabulary of values for the prediction variable Z. We determine the interpolation back-off order by looking at the number of values of each variable. We apply the following rule: the variable with the highest number of values observed in the training set is backed off first, then the variable with the next highest number of values, and so on. Typically, the class variable will be backed-off last according to this rule.

In Witten-Bell smoothing, the values of the interpolation coefficients are as follows: $\lambda(z_{1k}) = \frac{count(z_{1k})}{count(z_{1k}) + d \times |z{:}count(z, z_{1k}) > 0|}$. The weight of the relative frequency estimate based on a given context increases if the context has been seen more often in the training data and decreases if the context has been seen with more different values for the predicted variable z.

Looking at the form of our conditional probability tables, we can see that the major parameters are estimated directly based on the counts of the events in the training data. In addition, there are interpolation parameters (denoted by d above), which participate in computing the interpolation weights λ. The d parameters are hyper-parameters and we learn them on a development set of samples. We experimented with learning a single d parameter which is shared by all CPTs and learning multiple d parameters – one for every type of conditioning context in every CPT – i.e., each CPT has as many d parameters as there are back-off levels.

We place some restrictions on the Bayesian Networks learned, for closer correspondence with the discriminative models and for tractability: Every input variable node has the label node as a parent, and at most three parents per variable are allowed.

2.1.1 Structure Search Methodology

Our structure search method differs slightly from previously proposed methods in the literature (Heckerman, 1999; Pernkopf and Bilmes, 2005). The search space is defined as follows. We start with a Bayesian Network containing only the class variable. We denote by CHOSEN the set of variables already in the network and by REMAINING the set of unplaced variables. Initially, only the class variable Y is in CHOSEN and all other variables are in REMAINING. Starting from the current BN, the set of next candidate structures is defined as follows: For every unplaced variable R in REMAINING, and for every subset Sub of size at most two from the already placed variables in CHOSEN, consider adding R with parents $Sub \cup Y$ to the current BN. Thus the number of candidate structures for extending a current BN is on the order of m^3, where m is the number of variables.

We perform a greedy search. At each step, if the best variable B with the best set of parents $Pa(B)$ improves the evaluation criterion, move B from REMAINING to CHOSEN, and continue the search until there are no variables in REMAINING or the evaluation criterion can not be improved.

The evaluation criterion for BNs we use is classification accuracy on a development set of samples. Thus our structure search method is discriminative, in the terminology of (Grossman and Domingos, 2004; Pernkopf and Bilmes, 2005). It is very easy to evaluate candidate BN structures. The main parameters in the CPTs are estimated via the relative frequency estimator on the training set, as discussed in the previous section. We do not fit the hyper-parameters d during structure search. We fit these parameters only after we have selected a final BN structure. Throughout the structure search, we use a fixed value of 1 for d for all CPTs and levels of back-off. Therefore we are using generative parameter estimation and discriminative structure search. See Section 4 for discussion on how this method relates to previous work.

Notice that the optimal parameters of the conditional probability tables of variables already in the current BN do not change at all when a new variable is added, thus making update very efficient. After the stopping criterion is met, the hyper-parameters of the resulting BN are fit on the development set. As discussed in the previous subsection, we fit either a single or multiple hyper-parameters d. The fitting criterion for the generative Bayesian Networks is joint likelihood of the development set of samples with a Gaussian prior on the values $log(d)$. [1]

Additionally, we explore fitting the hyper-parameters of the Bayesian Networks by optimizing the conditional likelihood of the development set of samples. In this case we call the resulting models *Hybrid Bayesian Network* models, since they incorporate a number of discriminatively trained parameters. Hybrid models have been proposed before and shown to perform very competitively (Raina et al., 2004; Och and Ney, 2002). In Section 3.2 we compare generative and hybrid Bayesian Networks.

2.2 Discriminative Models

Discriminative models learn a conditional distribution $P_\theta(Y|\vec{X})$ or discriminant functions that discriminate between classes. Here we concentrate on conditional log-linear models. A simple example of such model is logistic regression, which directly corresponds to Naive Bayes but is trained to maximize the conditional likelihood. [2]

To describe the form of models we study, let us introduce some notation. We represent a tuple of nominal variables (X_1,X_2,…,X_m) as a vector of 0s and 1s in the following standard way: We map the tuple of values of nominal variables to a vector space with dimensionality the sum of possible values of all variables. There is a single dimension in the vector space for every value of each input variable X_i. The tuple (X_1,X_2,…,X_m) is mapped to a vector which has 1s in m places, which are the corresponding dimensions for the values of each variable X_i. We denote this mapping by Φ.

In logistic regression, the probability of a label $Y = y$ given input features $\Phi(X_1, X_2, \ldots, X_k) = \vec{x}$ is estimated as:

$$P(y|\vec{x}) = \frac{\exp\langle \vec{w_y}, \vec{x} \rangle}{\sum_{y'} \exp\langle \vec{w_{y'}}, \vec{x} \rangle}$$

There is a parameter vector of feature weights $\vec{w_y}$ for each label y. We fit the parameters of the log-linear model by maximizing the conditional likelihood of the training set including a gaussian prior on the parameters. The prior has mean 0 and variance σ^2. The variance is a hyper-parameter, which we optimize on a development set.

In addition to this simple logistic regression model, as for the generative models, we consider models with much richer structure. We consider more complex mappings Φ, which incorporate conjunctions of combinations of input variables. We restrict the number of variables in the combinations to three, which directly corresponds to our limit on number of parents in the Bayesian Network structures. This is similar to considering polynomial kernels of up to degree three, but is more general, because, for example, we can add only some and not all bigram conjunctions of variables. Structure search (or feature selection) for log-linear models has been done before e.g. (Della Pietra et al., 1997; McCallum, 2003). We devise our structure search methodology in a way that corresponds as closely as possible to our structure search for Bayesian Networks. The exact hypothesis space considered is defined by the search procedure for an optimal structure we apply, which we describe next.

2.2.1 Structure Search Methodology

We start with an initial empty feature set and a candidate feature set consisting of all input features: CANDIDATES={X_1,X_2,…,X_m}. In the course of the search, the set CANDIDATES may contain feature conjunctions in addition to the initial input features. After a feature is selected from the candidates set and added to the model, the feature is removed from CANDIDATES and all conjunctions of that feature with all input features are added to CANDIDATES. For example, if a feature conjunction $\langle X_{i_1}, X_{i_2}, \ldots, X_{i_n} \rangle$ is selected, all of its expansions of the form $\langle X_{i_1}, X_{i_2}, \ldots, X_{i_n}, X_i \rangle$, where X_i is not in the conjunction already, are added to CANDIDATES.

We perform a greedy search and at each step select the feature which maximizes the evaluation criterion, add it to the model and extend the set

[1]Since the d parameters are positive we convert the problem to unconstrained optimization over parameters γ such that $d = e^\gamma$.

[2]Logistic regression additionally does not have the sum to one constraint on weights but it can be shown that this does not increase the representational power of the model.

CANDIDATES as described above. The evaluation criterion for selecting features is classification accuracy on a development set of samples, as for the Bayesian Network structure search.

At each step, we evaluate all candidate features. This is computationally expensive, because it requires iterative re-estimation. In addition to estimating weights for the new features, we re-estimate the old parameters, since their optimal values change. We did not preform search for the hyper-parameter σ when evaluating models. We fit σ by optimizing the development set accuracy after a model was selected. Note that our feature selection algorithm adds an input variable or a variable conjunction with all of its possible values in a single step of the search. Therefore we are adding hundreds or thousands of binary features at each step, as opposed to only one as in (Della Pietra et al., 1997). This is why we can afford to perform complete re-estimation of the parameters of the model at each step.

3 Experiments

3.1 Problems and Datasets

We study two classification problems – prepositional phrase (PP) attachment, and semantic role labeling.

Following most of the literature on prepositional phrase attachment (e.g., (Hindle and Rooth, 1993; Collins and Brooks, 1995; Vanschoenwinkel and Manderick, 2003)), we focus on the most common configuration that leads to ambiguities: V NP PP. Here, we are given a verb phrase with a following noun phrase and a prepositional phrase. The goal is to determine if the PP should be attached to the verb or to the object noun phrase. For example, in the sentence: *Never* [*hang*]$_V$ [*a painting*]$_{NP}$ [*with a peg*]$_{PP}$, the prepositional phrase *with a peg* can either modify the verb *hang* or the object noun phrase *a painting*. Here, clearly, *with a peg* modifies the verb *hang*.

We follow the common practice in representing the problem using only the head words of these constituents and of the NP inside the PP. Thus the example sentence is represented as the following quadruple: [v:hang n_1:painting p:with n_2:peg]. Thus for the PP attachment task we have binary labels Att, and four input variables – v, n_1, p, n_2.

We work with the standard dataset previously used for this task by other researchers (Ratnaparkhi et al., 1994; Collins and Brooks, 1995). It is extracted from the the Penn Treebank Wall Street Journal data (Ratnaparkhi et al., 1994). Table 1 shows summary statistics for the dataset.

Task	Training	Devset	Test
PP	20,801	4,039	3,097
SRL	173,514	5,115	9,272

Table 1: Data sizes for the PP attachment and SRL tasks.

The second task we concentrate on is semantic role labeling in the context of PropBank (Palmer et al., 2005). The PropBank corpus annotates phrases which fill semantic roles for verbs on top of Penn Treebank parse trees. The annotated roles specify agent, patient, direction, etc. The labels for semantic roles are grouped into two groups, *core* argument labels and *modifier* argument labels, which correspond approximately to the traditional distinction between arguments and adjuncts.

There has been plenty of work on machine learning models for semantic role labeling, starting with the work of Gildea and Jurafsky (2002), and including CoNLL shared tasks (Carreras and Màrquez, 2005). The most successful formulation has been as learning to classify nodes in a syntactic parse tree. The possible labels are NONE, meaning that the corresponding phrase has no semantic role and the set of core and modifier labels. We concentrate on the subproblem of *classification* for core argument nodes. The problem is, given that a node has a core argument label, decide what the correct label is. Other researchers have also looked at this subproblem (Gildea and Jurafsky, 2002; Toutanova et al., 2005; Pradhan et al., 2005a; Xue and Palmer, 2004).

Many features have been proposed for building models for semantic role labeling. Initially, 7 features were proposed by (Gildea and Jurafsky, 2002), and all following research has used these features and some additional ones. These are the features we use as well. Table 2 lists the features. State-of-the-art models for the subproblem of classification of core arguments additionally use other features of individual nodes (Xue and Palmer, 2004; Pradhan et al., 2005a), as well as global features including the labels of other nodes in parse tree. Nevertheless it is interesting to see how well we can do with these 7 features only.

We use the standard training, development, and

Feature Types (Gildea and Jurafsky, 2002)
PHRASE TYPE: Syntactic Category of node
PREDICATE LEMMA: Stemmed Verb
PATH: Path from node to predicate
POSITION: Before or after predicate?
VOICE: Active or passive relative to predicate
HEAD WORD OF PHRASE
SUB-CAT: CFG expansion of predicate's parent

Table 2: Features for Semantic Role Labeling.

test sets from the February 2004 version of Propbank. The training set consists of sections 2 to 21, the development set is from section 24, and the test set is from section 23. The number of samples is listed in Table 1. As we can see, the training set size is much larger compared to the PP attachment training set.

3.2 Results

In line with previous work (Ng and Jordan, 2002; Klein and Manning, 2002), we first compare Naive Bayes and Logistic regression on the two NLP tasks. This lets us see how they compare when the generative model is making strong independence assumptions and when the two kinds of models have the same structure. Then we compare the generative and discriminative models with learned richer structures.

Table 3 shows the Naive Bayes/Logistic regression results for PP attachment. We list results for several conditions of training the Naive Bayes classifier, depending on whether it is trained as strictly generative or as a hybrid model, and whether a single or multiple hyper-parameters d are trained. In the table, we see results for generative Naive Bayes, where the d parameters are trained to maximize the joint likelihood of the development set, and for Hybrid Naive Bayes, where the hyper-parameters are trained to optimize the conditional likelihood. The column H-Params (for hyper-parameters) indicates whether a single or multiple d parameters are learned.

Logistic regression is more fairly comparable to Naive Bayes trained using a single hyper-parameter, because it also uses a single hyper-parameter σ trained on a development set. However, for the generative model it is very easy to train multiple weights d since the likelihood of a development set is differentiable with respect to the parameters. For logistic regression, we may want to choose different variances for the different types of features but the search would be prohibitively expensive. Thus we think it is also fair to fit multiple interpolation weights for the generative model and we show these results as well.

Model	**H-params**	**Test set acc**
Naive Bayes	1	81.2
Naive Bayes	9	81.2
Logistic regression	1	**82.6**
Hybrid Naive Bayes	1	81.2
Hybrid Naive Bayes	9	81.5

Table 3: Naive Bayes and Logistic regression PP attachment results.

As we can see from the table, logistic regression outperforms both Naive Bayes and Hybrid Naive Bayes. The performance of Hybrid Naive Bayes with multiple interpolation weights improves the accuracy, but performance is still better for logistic regression. This suggests that the strong independence assumptions are hurting the classifier. According to McNemar's test, logistic regression is statistically significantly better than the Naive Bayes models and than Hybrid Naive Bayes with a single interpolation weight ($p < 0.025$), but is not significantly better than Hybrid Naive Bayes with multiple interpolation parameters at level 0.05.

However, when both the generative and discriminative models are allowed to learn optimal structures, the generative model outperforms the discriminative model. As seen from Table 4, the Bayesian Network with a single interpolation weight achieves an accuracy of 84.6%, whereas the discriminative model performs at 83.8%. The hybrid model with a single interpolation weight does even better, achieving 85.0% accuracy. For comparison, the model of Collins & Brooks has accuracy of 84.15% on this test set, and the highest result obtained through a discriminative model with this feature set is 84.8%, using SVMs and a polynomial kernel with multiple hyper-parameters (Vanschoenwinkel and Manderick, 2003). The Hybrid Bayes Nets are statistically significantly better than the Log-linear model ($p < 0.05$), and the Bayes Nets are not significantly better than the Log-linear model. All models from Table 4 are significantly better than all models in Table 3.

For semantic role labelling classification of core arguments, the results are listed in Tables 5 and 6. We can see that the difference in performance between Naive Bayes with a single interpolation parameter d – 83.3% and the performance of Logistic regression – 91.1%, is very large. This shows that the independence assumptions are quite

Model	H-params	Test set acc
Bayes Net	1	84.6
Bayes Net	13	84.6
Log-linear model	1	83.8
Hybrid Bayes Net	1	**85.0**
Hybrid Bayes Net	13	84.8

Table 4: Bayesian Network and Conditional log-linear model PP attachment results.

Model	H-params	Test set acc
Naive Bayes	1	83.3
Naive Bayes	15	85.2
Logistic regression	1	**91.1**
Hybrid Naive Bayes	1	84.1
Hybrid Naive Bayes	15	86.5

Table 5: Naive Bayes and Logistic regression SRL classificaion results.

strong, and since many of the features are not sparse lexical features and training data for them is sufficient, the Naive Bayes model has no advantage over the discriminative logistic regression model. The Hybrid Naive Bayes model with multiple interpolation weights does better than Naive Bayes, performing at 86.5%. All differences between the classifiers in Table 5 are statistically significant at level 0.01. Compared to the PP attachment task, here we are getting more benefit from multiple hyper-parameters, perhaps due to the diversity of the features for SRL: In SRL, we use both sparse lexical features and non-sparse syntactic ones, whereas all features for PP attachment are lexical.

From Table 6 we can see that when we compare general Bayesian Network structures to general log-linear models, the performance gap between the generative and discriminative models is much smaller. The Bayesian Network with a single interpolation weight d has 93.5% accuracy and the log-linear model has 93.9% accuracy. The hybrid model with multiple interpolation weights performs at 93.7%. All models in Table 6 are in a statistical tie according to McNemar's test, and thus the log-linear model is not significantly better than the Bayes Net models. We can see that the generative model was able to learn a structure with a set of independence assumptions which are not as strong as the ones the Naive Bayes model makes, thus resulting in a model with performance competitive with the discriminative model.

Figures 2(a) and 2(b) show the Bayesian Networks learned for PP Attachment and Semantic Role Labeling. Table 7 shows the conjunctions

Model	H-params	Test set acc
Bayes Net	1	93.5
Bayes Net	20	93.6
Log-linear model	1	**93.9**
Hybrid Bayes Net	1	93.5
Hybrid Bayes Net	20	93.7

Table 6: Bayesian Network and Conditional log-linear model SRL classification results.

PP Attachment Model
⟨P⟩, ⟨P,V⟩, ⟨P,N_1⟩, ⟨P,N_2⟩
⟨N_1⟩,⟨V⟩, ⟨P,N_1,N_2⟩
SRL Model
⟨PATH⟩, ⟨PATH,PLEMMA⟩,⟨SUB-CAT⟩,⟨PLEMMA⟩
⟨HW,PLEMMA⟩,⟨PATH,PLEMMA,VOICE⟩
,⟨HW,PLEMMA,PTYPE⟩,⟨SUB-CAT,PLEMMA⟩
⟨SUB-CAT,PLEMMA,POS⟩,⟨HW⟩

Table 7: Log-linear models learned for PP attachment and SRL.

learned by the Log-linear models for PP attachment and SRL.

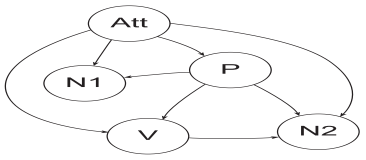
We should note that it is much faster to do structure search for the generative Bayesian Network model, as compared to structure search for the log-linear model. In our implementation, we did not do any computation reuse between successive steps of structure search for the Bayesian Network or log-linear models. Structure search took 2 hours for the Bayesian Network and 24 hours for the log-linear model.

To put our results in the context of previous work, other results on core arguments using the same input features have been reported, the best being 91.4% for an SVM with a degree 2 polynomial kernel (Pradhan et al., 2005a).[3] The highest reported result for independent classification of core arguments is 96.0% for a log-linear model using more than 20 additional basic features (Toutanova et al., 2005). Therefore our resulting models with 93.5% and 93.9% accuracy compare favorably to the SVM model with polynomial kernel and show the importance of structure learning.

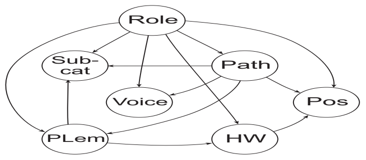
4 Comparison to Related Work

Previous work has compared generative and discriminative models having the same structure, such as the Naive Bayes and Logistic regression models (Ng and Jordan, 2002; Klein and Manning, 2002) and other models (Klein and Manning, 2002; Johnson, 2001).

[3]This result is on an older version of Propbank from July 2002.



(a) Learned Bayesian Network for PP attachment.



(b) Learned Bayesian Network for SRL.

Figure 2: Learned Bayesian Network structures for PP attachment and SRL.

Bayesian Networks with special structure of the CPTs – e.g. decision trees, have been previously studied in e.g. (Friedman and Goldszmidt, 1996), but not for NLP tasks and not in comparison to discriminative models. Studies comparing generative and discriminative models with structure learning have been previously performed ((Pernkopf and Bilmes, 2005) and (Grossman and Domingos, 2004)) for other, non-NLP domains. There are several important algorithmic differences between our work and that of (Pernkopf and Bilmes, 2005; Grossman and Domingos, 2004). We detail the differences here and perform an empirical evaluation of the impact of some of these differences.

Form of the generative models. The generative models studied in that previous work do not employ any special form of the conditional probability tables. Pernkopf and Bilmes (2005) use a simple smoothing method: fixing the probability of every event that has a zero relative frequency estimate to a small fixed ϵ. Thus the model does not take into account information from lower order distributions and has no hyper-parameters that are being fit. Grossman and Domingos (2004) do not employ a special form of the CPTs either and do not mention any kind of smoothing used in the generative model learning.

Form of the discriminative models. The works (Pernkopf and Bilmes, 2005; Grossman and Domingos, 2004) study Bayesian Networks whose parameters are trained discriminatively (by maximizing conditional likelihood), as representatives of discriminative models. We study more general log-linear models, equivalent to Markov Random Fields. Our models are more general in that their parameters do not need to be interpretable as probabilities (sum to 1 and between 0 and 1), and the structures do not need to correspond to Bayes Net structures. For discriminative classifiers, it is not important that their component parameters be interpretable as probabilities; thus this restriction is probably unnecessary. Like for the generative models, another major difference is in the smoothing algorithms. We smooth the models both by fitting a gaussian prior hyper-parameter and by incorporating features of subsets of cliques. Smoothing in (Pernkopf and Bilmes, 2005) is done by substituting zero-valued parameters with a small fixed ϵ. Grossman and Domingos (2004) employ early stopping using held-out data which can achieve similar effects to smoothing with a gaussian prior.

To evaluate the importance of the differences between our algorithm and the ones presented in these works, and to evaluate the importance of fitting hyper-parameters for smoothing, we implemented a modified version of our structure search. The modifications were as follows. For Bayes Net structure learning: (*i*) no Witten-Bell smoothing is employed in the CPTs, and (*ii*) no backoffs to lower-order distributions are considered. The only smoothing remaining in the CPTs is an interpolation with a uniform distribution with a fixed weight of $\alpha = .1$. For discriminative log-linear model structure learning: (*i*) the gaussian prior was fixed to be very weak, serving only to keep the weights away from infinity ($\sigma = 100$) and (*ii*) the conjunction selection was restricted to correspond to a Bayes Net structure with no features for subsets of feature conjunctions. Thus the only difference between the class of our modified discriminative log-linear models and the class of models considered in (Pernkopf and Bilmes, 2005; Grossman and Domingos, 2004) is that we do not restrict the parameters to be interpretable as probabilities.

The results shown in Table 8 summarize the results obtained by the modified algorithm on the two tasks. Both the generative and discriminative learners suffered a statistically significant (at level .01) loss in performance. Notably, the log-linear model for PP attachment performs worse than logistic regression with better smoothing.

PP Attachment Results		
Model	**H-params**	**Test set acc**
Bayes Net	0	82.8
Log-linear model	0	81.2
SRL Classification Results		
Model	**H-params**	**Test set acc**
Bayes Net	0	92.5
Log-linear model	0	92.7

Table 8: Bayesian Network and Conditional log-linear model: PP & SRL classification results using minimal smoothing and no backoff to lower order distributions.

In summary, our results showed that by learning the structure for generative models, we can obtain models which are competitive with or better than corresponding discriminative models. We also showed the importance of employing sophisticated smoothing techniques in structure search algorithms for natural language classification tasks.

Acknowledgements I would like to thank Bob Moore, Chris Manning, Andrew Ng, and Galen Andrew for useful discussions and the anonymous reviewers for their comments.

References

Xavier Carreras and Luís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Michael Collins and James Brooks. 1995. Prepositional attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16 – 23.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Nir Friedman and Moises Goldszmidt. 1996. Learning Bayesian Networks with local structure. In *Proceeding of UAI*, pages 252–262.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Joshua T. Goodman. 2001. A bit of progress in language modeling. In *MSR Technical Report MSR-TR-2001-72*.

Daniel Grossman and Pedro Domingos. 2004. Learning Bayesian Network classifiers by maximizing conditional likelihood. In *Proceedings of ICML*, pages 361—368.

David Heckerman. 1999. A tutorial on learning with Bayesian Networks. In *Learning in Graphical Models*. MIT Press.

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

Mark Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proceedings of ACL*.

Dan Klein and Christopher Manning. 2002. Conditional structure versus conditional estimation in NLP models. In *Proceedings of EMNLP*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of UAI*.

Andrew Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *NIPS 14*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Franz Pernkopf and Jeff Bilmes. 2005. Discriminative versus generative parameter and structure learning of Bayesian Network classifiers. In *Proceedings of ICML*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005a. Support Vector learning for semantic argument classification. *Machine Learning Journal*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of ACL*.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI*.

Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew McCallum. 2004. Classification with hybrid generative/discriminative models. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Workshop on Human Language Technology*.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL*.

Kristina Toutanova, Dan Klein, and Christopher D. Manning. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL*.

Bram Vanschoenwinkel and Bernard Manderick. 2003. A weighted polynomial information gain kernel for resolving prepositional phrase attachment ambiguities with Support Vector Machines. In *IJCAI*.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37,4:1085–1094.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.