Optimizing Local Probability Models for Statistical Parsing

Kristina Toutanova[1], Mark Mitchell[2], and Christopher D. Manning[1]

[1] Computer Science Department, Stanford University,
Stanford, CA 94305-9040, USA
{kristina,manning}@cs.stanford.edu
[2] CSLI, Stanford University,
Stanford, CA 94305, USA
markmitchell@fastmail.fm

Abstract. This paper studies the properties and performance of models for estimating local probability distributions which are used as components of larger probabilistic systems — history-based generative parsing models. We report experimental results showing that memory-based learning outperforms many commonly used methods for this task (Witten-Bell, Jelinek-Mercer with fixed weights, decision trees, and log-linear models). However, we can connect these results with the commonly used general class of deleted interpolation models by showing that certain types of memory-based learning, including the kind that performed so well in our experiments, are instances of this class. In addition, we illustrate the divergences between joint and conditional data likelihood and accuracy performance achieved by such models, suggesting that smoothing based on optimizing accuracy directly might greatly improve performance.

1 Introduction

Many disambiguation tasks in Natural Language Processing are not easily tackled by off-the-shelf Machine Learning models. The main challenges posed are the complexity of classification tasks and the sparsity of data. For example, syntactic parsing of natural language sentences can be posed as a classification task — given a sentence *s*, find a most likely parse tree *t* from the set of all possible parses of *s* according to a grammar *G*. But the set of classes in this formulation varies across sentences and can be very large or even infinite.

A common way to approach the parsing task is to learn a generative history-based model $P(s,t)$, which estimates the joint probability of a sentence s and a parse tree t [2]. This model breaks the complex (s,t) pair into pieces which are sequentially generated, assuming independence on most of the already generated structure. More formally, the general form of the history-based parsing model is $P(t) = \prod_{i=1}^{n} P(y_i|x_i)$. Here the parse tree is generated in some order, where every generated piece y_i (future) is conditioned on some context x_i (history).

The most important factors in the performance of such models are (*i*) the chosen generative model, including the representation of parse tree nodes, and (*ii*) the method for estimating the local probability distributions needed by the model. Due to the sparseness of NLP data, the method of estimating the local distributions $P(y_i|x_i)$ plays a very important role in building a good model. We will sometimes refer to this problem as smoothing.

The goals of the paper are three-fold: (*i*) to empirically evaluate the accuracy achieved by previously proposed and new local probability estimation models; (*ii*) to characterize the form of a kind of memory-based models that performed best in our study, showing their relation to deleted interpolation models; and (*iii*) to study the relationship among joint and conditional likelihood, and accuracy for models of this type.

While various authors have described several smoothing methods, such as using a deleted interpolation model [5], or a decision tree learner [13], or a maximum entropy inspired model [3], there has been a lack of comparisons of different learning methods for local decisions within a composite system. Because our ultimate goal here is to have good classifiers for choosing trees for sentences according to the rule $t = \arg\max_{t'} P(s,t')$, where the model $P(s,t')$ is a product of factors given by the local models $P(y_i|x_i)$, one can not isolate the estimation of local probabilities $P(y_i|x_i)$ as a stand-alone problem, choosing a model family and setting parameters to optimize the likelihood of test data. The bias-variance tradeoff may be different [9]. We find interesting patterns in the relationship between joint and conditional data likelihood and accuracy performance achieved by such compound models, suggesting that heavier smoothing is needed to optimize accuracy and that fitting a small number of parameters to optimize it directly might greatly improve performance.

The experimental study shows that memory-based learning outperforms commonly used methods for this task (Witten-Bell, Jelinek Mercer with fixed weights, decision trees, and log-linear models). For example, an error reduction of 5.8% in whole sentence accuracy is achieved by using memory-based learning instead of Witten-Bell, which is used in the state-of-the art model [5].

2 Memory-Based and Deleted Interpolation Models

In this section we demonstrate the relationship between deleted interpolation models and a class of memory-based models that performed best in our study.

2.1 Deleted Interpolation Models

Deleted interpolation models estimate the probability of a class y given a feature vector (context) of n features, $P(y|x_1,\ldots,x_n)$, by linearly combining relative frequency estimates based on subsets of the full context $(x_1,\ldots,x_n)$, using statistics from lower-order distributions to reduce sparseness and improve the estimate. To write out an expression for this estimate, let us introduce some notation. We will denote by S_j subsets of the set $\{1,\ldots,n\}$ of feature indices. S_j can take on 2^n values ranging from the empty set to the full set $\{1,\ldots,n\}$. We will denote by X_S the tuple of feature values of X for the features whose indices are in S. For example $X_{\{1,2,3\}} = (x_1,x_2,x_3)$. For convenience, we will add another set, denoted by $*$, which we will use to include in the interpolation the uniform distribution $P(y) = \frac{1}{V}$, where V is the number of possible classes y. The general form of estimate is then:

$$\tilde{P}(y|X) = \sum_{S_i \subseteq \{1,\ldots,n\} \vee S_i = *} \lambda_{S_i}(X)\hat{P}(y|X_{S_i}) \tag{1}$$

Here $\hat{P}$ are relative frequency estimates and $\hat{P}(y|X_*) = \frac{1}{V}$ by definition. The interpolation weights λ are shown to depend on the full context $X = (x_1, \ldots, x_n)$ as well as the specific subset S_i of features. In practice parameters as general as that are never estimated. For strictly linear feature subsets sequences, methods have been proposed to fit the parameters by maximizing the likelihood of held-out data through EM while tying parameters for contexts having equal or similar counts.[3]

2.2 (A Kind of) Memory-Based Learning Models

We will show that a broad class of memory-based learning methods have the same form as Equation 1 and are thus a subclass of deleted interpolation models. While [18] have noted that memory-based and back-off models are similar in the way they use counts and in the way they specify abstraction hierarchies among context subsets, the exact nature of the relationship is not made precise. They emphasize the case of 1-nearest neighbor and show that it is equivalent to a special kind of strict back-off (non-interpolated) model. Our experimental results suggest that a number of neighbors K much larger than 1 works best for local probability estimation in parsing models. The exact form of the interpolation weights λ as dependent on contexts and their counts is therefore crucial for combining more specific and more general evidence. We will look at memory-based learning models determined by the following parameters:

– K, the number of nearest neighbors.
– A distance function $\Delta(X, X')$ between feature vectors. This function should depend only on the *positions* of matching/mis-matching features.
– A weighting function $w(X, X')$, which is the weight of neighbor X' of X. We will assume that the weight is a function of the distance, i.e. $w(X, X') = w(\Delta(X, X'))$.

Let us denote by $N_K(X)$ the set of K nearest neighbors of X. The probability of a class y given X is estimated as:

$$\tilde{P}(y|X) = \frac{\sum_{X' \in N_K(X)} w(\Delta(X, X'))\delta(y, y')}{\sum_{X' \in N_K(X)} w(\Delta(X, X'))} \tag{2}$$

Here y' is the label of the neighbor X', and $\delta(y, y') = 1$ iff $y = y'$, and 0 otherwise. For nominal attributes, as always used in conditioning contexts for natural language parsers, the distance function commonly distinguishes only between matches and mismatches on feature values, rather than specifying a richer distance between values. We

[3] When not limited to linear subsets sequences, it is possible to optimize tied parameters, but EM is difficult to apply and we are not aware of work trying to optimize interpolation parameters for models of this more general form.

will limit our analysis to this case as specified in the conditions above. In the majority of applications of k-NN to natural language tasks, simple distance functions have been used [6].[4]

The distance function $\Delta(X, X')$ will take on one of 2^n values depending on the indices of the matching features between the two vectors. In practice we will add V artificial instances to the training set, one of each class (to avoid zero probabilities). These instances will be at an additional distance value δ_{smooth} which will normally be larger that the other distances. We require that the distance $\Delta(X, X')$ be no smaller than $\Delta(X, X'')$ if X'' matches X on a superset of the attributes on which X' matches.

The commonly used overlap distance function, $\Delta(X, X') = \sum_{i=1}^{n} w_i \delta(x_i, x'_i)$, satisfies these constraints. Every feature has an importance weight $w_i \geq 0$. This is the distance function we have used in our experiments, but it is more restrictive than the general case for which our analysis holds, because it has only $n+1$ parameters — the w_i and δ_{smooth}. The general case would require $2^n + 1$ parameters.

We go on to introduce one last bit of notation. We will say that the schema S of an instance X' with respect to an instance X is the set of feature indices on which the two instances match. (We are herer using similar terminology to [18]). It is clear that the distance $\Delta(X, X')$ depends only on the schema S of X' with respect to X. The same holds true for the weight of X' with respect to X. We can therefore think of the K nearest neighbors as groups of neighbors that have the same schema. Let us denote by $S_K(X)$ the set of schemata of the K nearest neighbors of X. We assume that instances in the same schema are either all included or excluded from the nearest neighbors set. The same assumption has commonly been made before [18]. We have the following relationships between schemata $S' \leq S$ if the schema S' is more specific than S, i.e. the set of feature indices S' is a superset of the set S. We will use $S' \prec S$ for immediate precedence, i.e. $S' \prec S$ iff $S' \leq S$ and there are no schemata between the two in the ordering. We can rearrange Equation 2 in terms of the participating schemata and then after an additional re-bracketing, we obtain the same form as Equation 1.

$$\tilde{P}(y|X) = \sum_{S_j \in S_K(X)} \lambda_{S_j}(X) \hat{P}(y|X_{S_j}) \tag{3}$$

The interpolation coefficients have the form:

$$\lambda_{S_j}(X) = \frac{(w(\Delta(S_j)) - \sum_{S_j \prec S'_j, S'_j \in S_K(X)} w(\Delta(S'_j)))}{Z(X)} c(X_{S_j}) \tag{4}$$

$$Z(X) = \sum_{S_j \in S_K(X)} \left(w(\Delta(S_j)) - \sum_{S_j \prec S'_j, S'_j \in S_K(X)} w(\Delta(S'_j)) \right) c(X_{S_j}) \tag{5}$$

This concludes our proof that memory-based models of this type are a subclass of deleted interpolation models. It is interesting to observe the form of the interpolation

[4] Richer distance functions have been proposed and shown to be advantageous [18, 12, 8]. However, such distance functions are harder to acquire and using them raises significantly the computational complexity of applying the k-NN algorithm. When simple distance functions are used, clever indexing techniques make testing a constant time operation.

coefficients. We can notice that they depend on the total number of instances matching the feature subset as is usually true of other linear subsets deleted interpolation methods such as Jelinek-Mercer smoothing and Witten-Bell smoothing. However they also depend on the counts of more general subsets as seen in the denominator. The different counts are weighted according to the function w.

In practice the most widely used deleted interpolation models exclude some of the feature subsets and estimates are interpolated from a linear feature subsets order. These models can be represented in the form:

$$\tilde{P}(y|x_1,\ldots,x_n) = \lambda_{x_1,\ldots,x_n}\hat{P}(y|x_1,\ldots,x_n) + (1-\lambda_{x_1,\ldots,x_n})\tilde{P}(y|x_1,\ldots,x_{n-1}) \quad (6)$$

The recursion is ended with the uniform distribution as above. Memory-based models will be subclasses of deleted interpolation models of this form if we define $\Delta(S) = \Delta(\{1,\ldots,i\})$, where i is the largest numbers such that $\{1,\ldots,i\} \geq S$. If such i does not exist $\Delta(S) = \Delta(\{\})$ or $\Delta(*)$ for the artificial instances.

3 Experiments

We investigate these ideas via experiments in probabilistic parse selection from among a set of alternatives licensed by a hand-built grammar in the context of the newly developed Redwoods HPSG treebank [14]. HPSG (Head-driven Phrase Structure Grammar) is a modern constraint-based lexicalist (unification) grammar, described in [15].

The Redwoods treebank makes available syntactic and semantic analyses of much greater depth than, for example, the Penn Treebank. Therefore there are a large number of features available that could be used by stochastic models for disambiguation. In the present experiments, we train generative history-based models for derivation trees. The derivation trees are labeled via the derivation rules that build them up; an example is shown in Figure 1. All models use the 8 features shown in Figure 2. They estimate the probability $P(expansion(n)|history(n))$, where *expansion* is the tuple of node labels of the children of the current node and *history* is the 8-tuple of feature values. The results we obtain should be applicable to Penn Treebank parsing as well, since we use many similar features such as grand-parent information and build similar generative models.

The accuracy results we report are averaged over a ten-fold cross-validation on the data set summarized in Table 1. Accuracy results denote the percentage of test sentences for which the highest ranked analysis was the correct one. This measure scores whole sentence accuracy and is therefore stricter than the labelled precision/recall measures, and more appropriate for the task of parse selection.[5]

3.1 Linear Feature Subsets Order

In this first set of experiments, we compare memory-based learning models restricted to linear order among feature subsets to deleted interpolation models using the same

[5] Therefore we should expect to obtain lower figures for this measure compared to labelled precision/recall. As an example, the state of the art unlexicalized parser [11] achieves 86.9% F measure on labelled constituents and 30.9% exact match accuracy.

Table 1. Annotated corpus used in experiments: The columns are, from left to right, the total number of sentences, average length, average lexical ambiguity (number of lexical entries per token), average structural ambiguity (number of parses per sentence), and the accuracy of choosing at random

sentences	length	lex ambiguity	struct ambiguity	random baseline
5312	7.0	4.1	8.3	25.81%

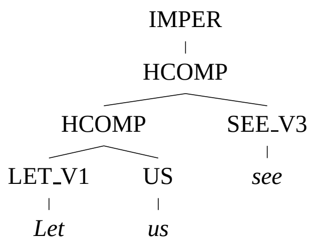


Fig. 1. Example of a Derivation Tree

No.	Name	Example
1	Node Label	*HCOMP*
2	Parent Node Label	*HCOMP*
3	Node Direction	*left*
4	Parent Node Direction	*none*
5	Grandparent Node Label	*IMPER*
6	Great Grandparent Label	*yes*
7	Left Sister Node Label	*HCOMP*
8	Category of Node	*verb*

Fig. 2. Features over derivation trees

linear subsets order. The linear interpolation sequence was the same for all models and was determined by ordering the features of the history by gain-ratio. The resulting order was: $1, 8, 2, 3, 5, 4, 7, 6$ (see Table 2). Numerous methods have been proposed for estimation of parameters for linearly interpolated models.[6] In this section we survey the following models:

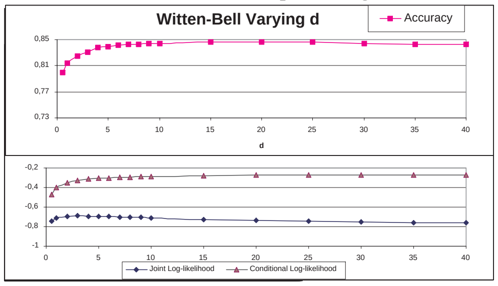
Jelinek Mercer with a fixed interpolation weight λ for the lower-order model (and $1 - \lambda$ for the higher-order model). This is a model of the form of Equation 6, where the interpolation weights do not depend on the feature history. We report test set accuracy for varying values of λ. We refer to this model as JM.

Witten-Bell smoothing [17] uses as an expression for the weights : $\lambda(x_1, \ldots, x_i) = \frac{c(x_1,\ldots,x_i)}{c(x_1,\ldots,x_i)+d\times|y:c(y,x_1,\ldots,x_i)>0|}$. We refer to this model as WBd. The original Witten-Bell smoothing[7] is the special case with $d = 1$, but use of an additional parameter d which multiplies the number of observed outcomes in the denominator is commonly used in some of the best-performing parsers and named-entity recognizers [1, 5].

Memory-based models restricted to linear sequence, with varying weight function and varying values of K. The restriction to linear sequence is obtained by defining the distance function to be of the special form described at the end of section 2. We define the distance function as follows for subsets of the linear generalization sequence: $\Delta(\{1, \ldots, n\}) = 0, \cdots, \Delta(\{\}) = n, \Delta(*) = n+1$. We implemented several weighting methods, including inverse, exponential,information gain, and gain-ratio. The weight functions inverse cubed(INV3) and inverse to the fourth (INV4) worked best. They are

[6] In addition to models of the form of Equation 6 there are models that use modified distributions (not the relative frequency). Comparison to these other good models (e.g., forms of Kneser-Ney and Katz smoothing [4] is not the subject of this study and would be an interesting topic for future research.

[7] *Method C*, also used in [4].


Witten-Bell Varying d
Accuracy
0,85
0,81
0,77
0,73
0
5
10
15
20
25
30
35
40
d
-0,2
-0,4
-0,6
-0,8
-1
Joint Log-likelihood
Conditional Log-likelihood