Expected-Frequency Interpolation

Eugene Charniak

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-96-37
December 1996

Expected-Frequency Interpolation*

Eugene Charniak
Department of Computer Science, Brown University

December 30, 1996

Abstract

Expected-frequency interpolation is a technique for improving the performance of deleted interpolation smoothing. It allows a system to make finer-grained estimates of how often one would expect to see a particular combination of events than is possible with traditional frequency interpolation. This allows the system to better weigh the emphasis given to the various probablity distributions being mixed. We show that more traditional frequency interpolation, based solely on the frequency of conditioning events, can lead to some anomalous results. We then show that while the equations for expected-frequency interpolation are not exact, they are close, depending on how well some seemingly reasonable assumptions hold. We then present an experiment in which the introduction of expected-frequence interpolation to a statistical parsing system improved performance by .4% with essentially no extra work, and essentially no change in the workings of the system. We also note that even before the change, the system in question was the top performer at its task, so a .4% improvement was well worth obtaining.

1 Introduction

Sparse data problems in statistically-based systems arise from the fact that the system's training data hardly ever covers all of the cases the system will

*This research was supported in part by NSF grant IRI-9319516 and by ONR grant N0014-96-1–0549.

encounter. A statistical tagger encounters a word it has never seen before, a parser sees a word used as the head of a noun-phrase that previously was only seen as an adjective. Thus most statistics used in such systems must be *smoothed*, the sharing of probability mass from cases that have been observed to (hopefully) similar cases that have not.

One standard scheme for such sharing is *interpolation*, and in particular *deleted interpolation*. In deleted interpolation one estimates a conditional probability by linearly interpolating several probabilities, each one deleting some of the conditioning events of the probability to be estimated.

The canonical example of this occurs in trigram models of language (e.g., [1]), where the basic probability is the probability of a word given the two previous words in the text, $p(w_i \mid w_{i-2}, w_{i-1})$, where w_i is the ith word in the text. Since the number of word triples in natural languages is very large, the training data inevitably does not include many plausible word triples, and thus $p(w_i \mid w_{i-2}, w_{i-1})$ is often estimated using the deleted interpolation formula:

$$\begin{aligned} p(w_i \mid w_{i-2}, w_{i-1}) &= \lambda_1(w_{i-2}, w_{i-1})\hat{p}(w_i \mid w_{i-2}, w_{i-1}) \\ &\quad +\lambda_2(w_{i-2}, w_{i-1})\hat{p}(w_i \mid w_{i-1}) \\ &\quad +\lambda_3(w_{i-2}, w_{i-1})\hat{p}(w_i). \end{aligned} \tag{1}$$

Here, as elsewhere in this paper we use $\hat{p}(s)$ to denote a maximum likelihood estimation of $p(s)$ empirically obtained from a corpus. As a bit of terminology we distinguish between the "conditioning events" (in $p(w_i \mid w_{i-2}, w_{i-1})$ they would be w_{i-2} and w_{i-1}) and the "conditioned event" (in $p(w_i \mid w_{i-2}, w_{i-1})$ it would be w_i).

Note that if Equation 1 is to be a probability distribution it must be the case that for all w_{i-2} and w_{i-1},

$$\lambda_1(w_{i-2}, w_{i-1}) + \lambda_2(w_{i-2}, w_{i-1}) + \lambda_3(w_{i-2}, w_{i-1}) = 1. \tag{2}$$

Also note that the λs may be, in general, functions of the conditioning events. This is useful because it allows the system to use more of the more specific statistics in situations where it has reason to believe that the data is more plentiful, and less of them where it is not. For example, if the last two words are "of the," or "is a," the system might have better confidence in $\hat{p}(w_i \mid w_{i-2}, w_{i-1})$ than it would if the last two words are, say "baked zucchini." Naturally, if we are to enforce something like Equation 2 the λs cannot be a function of the conditioned random variable (in the above, w_i).

More generally, one typically adjusts the λ values as functions of the frequency of the conditioning events. The more such events, the higher the λs for the most conditioned probabilities. So, if $\mid w_{i-2}, w_{i-1} \mid$ is the frequency of the these two words appearing in this order, then we would use the following smoothing equation:[1]

$$\begin{aligned} p(w_i \mid w_{i-2}, w_{i-1}) &= \lambda_1(\mid w_{i-2}, w_{i-1} \mid)\hat{p}(w_i \mid w_{i-2}, w_{i-1}) \\ &\quad +\lambda_2(\mid w_{i-2}, w_{i-1} \mid)\hat{p}(w_i \mid w_{i-1}) \\ &\quad +\lambda_3(\mid w_{i-2}, w_{i-1} \mid))\hat{p}(w_i). \end{aligned} \tag{3}$$

We call smoothing equations of this sort *frequency interpolation*. A good example of frequency interpolation can be found in [1]. An example of deleted interpolation used for tagging models can be found in [6], which also discusses the use of frequency interpolation.

While deleted interpolation, and more specifically, frequency interpolation works reasonably well for trigram language modeling, we suspect that this is due in part to the particularly simple structure of the problem. To make this more concrete, in section two we outline a frequency interpolation model used for parsing, and show how frequency interpolation can lead to anomalous results in many cases. Section three outlines our solution to this problem, while section four gives some empirical results.

2 Frequency Interpolation in Parsing

In [2] we describe a program that parses using a language model. The program assigns every sentence a probability that is the sum of the probabilities of all of the sentence's possible parses (or at least all that the model constructs). This becomes a parsing model when we have it choose as the correct parse of the sentence that with the highest probability.

The details of this model are not important here, except for a few. First, every phrase in the parse is assigned a "head" lexical item. This is the word of the phrase that is considered the most important. So the head of

[1]Strictly speaking, λ_2 and λ_3 should also be a function of $\mid w_{i-1} \mid$ since the relative weight of λ_2 with respect to λ_3 should depend just on the frequence of the last word, and is independent of the second to last. Similarly, λ_3 should be a function of both counts since the λs must sum to 1. We ignore these fine points.

a noun phrase is typically its rightmost noun, e.g., the head of "a lengthy conversation about politics" would be "conversation." Perhaps less intuitive, but still reasonable, the head of a prepositional phrase, pp, is the preposition (e.g., the head of "about politics" is "about").

Second, when calculating the probability of a sentence according to a particular parse, the probability of a head of a phrase is conditioned on the head of the phrase just above it (according to that parse). For example, consider the following sentence:

> They started a lengthy conversation about politics.

Parsing this sentence requires, among other things, deciding if the pp "about politics" attaches to the noun "conversation" or the verb "started." A key probability in this decision would be the the probability of "about" given the head of the higher phrase. In the parse that had the pp attached to "conversation" the probability would be $p(\text{about} \mid \text{conversation})$ whereas in the parse in which the pp is attached to "started" the relevant probability is $p(\text{about} \mid \text{started})$. The reader might note that these probabilities are the same ones used by Hindle and Rooth [4] in their early work on pp attachment.

More generally we say that the probability of the (sub)head s of a phrase of type t (e.g., pp) according to a parse in which the phrase is attached to a parent headed by h is $p(s \mid h, t)$, and this probability is approximated using deleted interpolation according to the following formula.

$$p(s \mid h, t) = \lambda_1(h, t)\hat{p}(s \mid h, t) + \lambda_2(h, t)p(s \mid t) \qquad (4)$$

As before $\hat{p}(s \mid h, t)$ is obtained from the training corpus. The second probability, $p(s \mid t)$ also has a large component that is obtained from counting what happens in the training corpus, but includes as well a mechanism to assign probabilities to word-tag combinations that have not appeared in the training data, (Actually, the smoothing equation used in [2] is somewhat more complicated than Equation 4, but these complications are not relevant to this paper, and we ignore them.)

Next, we need to discuss how values are assigned to $\lambda_1(h, t)$. We do not need to worry about $\lambda_2(h, t)$ since the two λs must sum to one for all p, t, and thus λ_2 is completely determined by λ_1.

As we noted above, using the frequency of the conditioning events seems like a reasonable thing to do, so that is what we did. Thus we let $\lambda_1(h, t) =$

$\lambda_1(\mid h, t \mid)$ where again $\mid h, t \mid$ is the number of times a constituent headed by p has a subconstituent of type t. In our running example of pp attachment, this would be the number of times the word "conversation" (or "started") headed a constituent with a pp in it.

This seems reasonable, but it can lead to funny problems. To see this, consider the following example:

> We had a tête-à-tête during a walk amongst the roses.

Here we assume that the words "tête-à-tête" and "amongst" are words that do not appear in the training data, and that all of the others words do appear.

The point of this example is the following odd fact: when we use frequency interpolation, Equation 4 pushes the parser toward attaching "amongst" to "tête-à-tête" rather than to "walk," despite the fact that we have no statistics on either combination. The reason is simple. Whatever "amongst" attaches to, $\hat{p}(s \mid h, t)$ is zero, since this is obtained directly from the training data, and we are assuming that s, here "amongst," does not appear in the data. Now this is what the second term of Equation 4 is designed to handle. But, and this is the crucial point, $\lambda_2(\mid h, t \mid)$ is significantly larger attaching "amongst" to "tête-à-tête" than to "walk." This is simply because tête-à-tête is unknown, and thus the frequencey $\mid$ tête-à-tête, $\mathsf{pp} \mid= 0$, so the λ_1 is zero (making $\lambda_2 = 1$) while when computing the probability for attaching to "walk" presumably the frequency $\mid$ walk, $\mathsf{pp} \mid$ is non zero, and may even be reasonably high. Thus, since $p(\text{amongst} \mid \mathsf{pp})$ is the same for either attachment, the attachment with the larger $\lambda_2(\mid h, t \mid)$ has the higher value, and this, as we have just argued, is "tête-à-tête."

More generally, Equation 4 causes unknown or low-frequency words to try to attach to other unknown words, and if there are no other unknown words in the vicinity, to try to attach to the least common word, the better to make λ_1 as small as possible. This is the anomalous situation to which we earlier referred.

3 Expected-Frequency Interpolation

To get a better perspective on exactly where the problem lies, note that in some cases the behavior shown in the last section is reasonable. In particular, suppose we have a very common word w that might attach at two possible points, whose heads are a and b. Assume that a is quite common, and b is quite rare, but suppose that w has never been seen to attach to either a or b. In this case it does seem reasonable to prefer b. If both w and a are common, then the fact that we have never seen them attached is significant, and may well mean that w typically would not go under a. In the case of b, the uncommon word, the lack of previous cases is not statistically significant, and thus w may well be a fitting argument.

The difference between this case and the one outlined in the last section is instructive — what we really care about in assigning λs is not how often the system observes the conditioning events, but how often, everything else being equal, it might expect to see see the conditioning events *and the conditioned event* s.

In this paper we propose handling both situations, the reasonable one just outlined, and the anomalous one of the last section, by replacing the frequency of the conditioning events $\mid h, t \mid$ with a a measure of it and the conditioned event, namely

$$f(s, h, t) \equiv \mid h, t \mid \cdot \hat{p}(s \mid t) \tag{5}$$

If both h and s are common, then $f(h, t, s)$ is large, but, as in the "amongst" example, if s is uncommon, them $f(h, t, s)$ is small, and λ_1 is small as well. Substituting f in as the basis for determining the λ values in Equation 4 gives us the following equation:

$$p(s \mid h, t) = \lambda_1(f(s, h, t))\hat{p}(s \mid h, t) + \lambda_2(f(s, h, t))p(s \mid t) \tag{6}$$

Generically we call the interpolation method exhibited in Equation 6 (based upon frequency functions like f) *expected-frequency interpolation* since, f as defined in Equation 5 is an estimate of how often one expects to see s, h, t if the probability of s is independent of h.

This seems reasonable, except for one thing. We noted earlier that to ensure the λ values summing to one, it was not permissible to have them depend on s. We are now breaking this rule. What are the consequences? As we show in the rest of this section, under certain reasonable assumptions the consequences are not great. However, showing this involves a fair amount of formula crunching, and those willing to accept this on faith are encouraged to skip to the next section where we give some experimental evidence on the efficacy of the method.

In what follows we take Equation 4 as a starting point. In doing so, we make no use of the particular meanings we have given to the various terms therein, except to argue the reasonableness of certain assumptions. Thus our argument is of the form, if the following assumptions are reasonable, then ... Also, our argument holds for interpolation equations where we are interpolating between more than the two probabilities of Equation 4. Showing this is reasonably straight forward except the notation becomes somewhat cumbersome, so we do not do it here.

First, for reasons that will become clearer later in this section, rather than deal directly with $\hat{p}(s \mid t)$ in Equation 5 we discretize the values. More formally, we define a sequence of v value ranges for $\hat{p}(s \mid t)$ $\{V_i | i = 1, v\}$ where $V_1 = 0$, $V_{i+1} > V_i$, and $V_v = 2$. (Here 2 is just a number larger than 1 so that the final value is larger than any possible value of $\hat{p}(s \mid t)$.) We then partition $\hat{p}(s \mid t)$ into these buckets with the function g.

$$g(s,t) = i \text{ iff } V_i \leq \hat{p}(s \mid t) < V_{i+1} \tag{7}$$

We now introduce this discrete version of $\hat{p}(s \mid t)$ into our equations.

$$\begin{aligned} p(s \mid h,t) &= \sum_{i=1,v} p(g(s,t) = i \mid h,t) p(s \mid h,t,g(s,t) = i) & (8) \\ &= \sum_{i=1,v} p(i \mid h,t) p(s \mid h,t,i) & (9) \end{aligned}$$

Here Equation 8 just introduces i from Equation 7, and Equation 9 is the same as 8 except we have dropped the explicit $g(s,t) = i$, as we do in the rest of this paper.

In essence our mathematical goal at this point is to first introduce the deleted interpolation idea into Equation 9, and then, as far as possible, remove all occurrences of i outside of the λ functions.

$$\begin{aligned} p(s \mid h,t) &= \sum_{i=1,v} p(i \mid h,t) p(s \mid h,t,i) & (10) \\ &= \sum_{i=1,v} p(i \mid h,t) & (11) \\ &\qquad \cdot \{\lambda_1(h,t,i)\hat{p}(s \mid h,t,i) + \lambda_2(h,t,i)p(s \mid t,i)\} \end{aligned}$$

In Equation 11 we can make our λs functions of i since the latter now appears as a conditioning event in our distributions. Indeed, we could stop here if we chose, since this equation achieves what we want in terms of allowing λs to take $\hat{p}(s \mid t)$ into account. To actually implement this, first we would estimate empirically $\hat{p}(i \mid h, t)$ from our corpus, and then assume

$$p(i \mid h, t) = \hat{p}(i \mid h, t). \tag{12}$$

That is, assume that our empirically obtained values are close enough to the truth. In what follows we make this assumption although obviously it is not completely justified. On the other hand, we are assuming that $\hat{p}(s \mid h, t)$ is close enough to the true distribution to make it worth using in our deleted interpolation formula, and this probability estimate faces much sparser data than does $\hat{p}(i \mid h, t)$ assuming, as we will, that the number of i values, v, is quite low, say, ten or twenty.

However, using Equation 11 would require us to completely revise the probabilities we calculate. For example, rather than computing $\hat{p}(s \mid h, t)$, Equation 11 requires $\hat{p}(s \mid h, t, i)$. As we now show, at the price of a few more assumptions we can keep the form of our probabilities exactly the same.

$$\begin{aligned}
p(s \mid h, t) &= \sum_{i=1,v} \lambda_1(h,t,i) p(i \mid h,t) \hat{p}(s \mid h,t,i) \\
&\qquad + \lambda_2(h,t,i) p(i \mid h,t) p(s \mid t,i) && (13) \\
&= \sum_{i=1,v} \lambda_1(h,t,i) \hat{p}(i \mid h,t) \frac{\hat{p}(s,h,t,i)}{\hat{p}(h,t,i)} && (14) \\
&\qquad + \lambda_2(h,t,i) p(i \mid h,t) \frac{p(s,t,i)}{p(t,i)} \\
&= \sum_{i=1,v} \lambda_1(h,t,i) \hat{p}(i \mid h,t) \frac{\hat{p}(s,h,t)\hat{p}(i \mid s,h,t)}{\hat{p}(h,t)\hat{p}(i \mid h,t)} && (15) \\
&\qquad + \lambda_2(h,t,i) p(i \mid h,t) \frac{p(s,t) p(i \mid s,t)}{\hat{p}(t) p(i \mid t)} \\
&= \sum_{i=1,v} \lambda_1(h,t,i) \hat{p}(i \mid h,t) \frac{\hat{p}(s \mid h,t)\hat{p}(i \mid s,h,t)}{\hat{p}(i \mid h,t)} && (16) \\
&\qquad + \lambda_2(h,t,i) p(i \mid h,t) \frac{p(s \mid t) p(i \mid s,t)}{p(i \mid t)} \\
&= \sum_{i=1,v} \lambda_1(h,t,i) \hat{p}(s \mid h,t) \hat{p}(i \mid s,h,t) && (17)
\end{aligned}$$

$$+\lambda_2(h,t,i)p(s \mid t)p(i \mid s,t)\frac{p(i \mid h,t)}{p(i \mid t)}$$

Now we note from Equation 7 that i is a deterministic function of s, t and thus

$$\hat{p}(i \mid s,h,t) = p(i \mid s,t) = \begin{cases} 1 \Leftrightarrow g(s,t) = i \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

Thus there is only one value of i for which the quantity under the summation in Equation 17 is non-zero, $i = g(s,t)$. Thus we drop consideration of the other i values, and indicate this by dropping the summation and replace i in Equation 17 with $g(s,t)$, giving us:

$$\begin{aligned} p(s \mid h,t) &= \lambda_1(h,t,g(s,t))\hat{p}(s \mid h,t) \\ &\quad +\lambda_2(h,t,g(s,t))p(s \mid t)\frac{p(g(s,t) \mid h,t)}{p(g(s,t) \mid t)} \end{aligned} \tag{19}$$

At this point we have an equation remarkably like Equation 4 except the λ values are a function of i, the discrete indicator of $p(s \mid t)$ (which was our goal in this entire operation), and, less desirably, a fraction at the end

$$\frac{p(g(s,t) \mid h,t)}{p(g(s,t) \mid t)} \tag{20}$$

We now turn our attention to this fraction.

To get rid of this we make our second assumption (Equation 12 is our first).

$$p(g(s,t) \mid h,t) = p(g(s,t) \mid t). \tag{21}$$

This is an independence assumption as we assume that the probability of $g(s,t)$ is independent of h given t. As before, the reasonableness of the assumption depends in part on the coarseness of our discretization of $\hat{p}(s \mid t)$ incorporated into the function g. If this is not very course, (i.e., if v is very large) then unless two words have exactly the same $\hat{p}(s \mid t)$ then they will fall into different buckets so each bucket would have exactly one word in it. If this is the case then

$$p(g(s,t) \mid h,t) = p(s \mid h,t). \tag{22}$$

Thus for large v our independence assumption boils down to the assumption that the two distributions, $p(s \mid h,t)$ and $p(s \mid t)$ are the same, in which case there is no point in the interpolation at all.

On the other hand, if v is small, say, the 10 to 20 we talked about earlier, then the independence assumption looks plausible. In effect it is saying that if we know the part of speech t of s, then knowing h, the head of the phrase above s, does not tell us much more about the gross distribution of $\hat{p}(s \mid t)$. We can think of some cases where this is false, but on the whole this would appear to be a reasonable assumption.[2]

At any rate, Equations 12 and 21 are the assumption we have been warning about all along, and we make them. Once we do so we get the equation we have been after.

$$p(s \mid h, t) = \lambda_1(h, t, g(s, t))\hat{p}(s \mid h, t) + \lambda_2(h, t, g(s, t))p(s \mid t) \tag{23}$$

Then to get the frequence-based interpolation form of the equation we just make the λs a function of $f(h, t, g(s, t))$ giving us:

$$p(s \mid h, t) = \lambda_1(f(h, t, g(s, t)))\hat{p}(s \mid h, t) + \lambda_2(f(h, t, g(s, t)))p(s \mid t) \tag{24}$$

Lastly, note that in practice one typically does not expect λs to be continuous functions of their arguments, but rather discrete function. Or to put it another way, rather than use the continuous values of f we break its range up into buckets. The reason is that the best way to actually get the λ values is to train them on some held-out data using the expectation-maximization algorithm. Making the domain discrete is the most obvious way to avoid the problem of sparse data for the λs themselves. Once one uses discrete domains for the λs it is reasonable to back off from the discretization of $\hat{p}(s \mid t)$ and just use the fact once we compute the expected frequency it is discretized at that point. This is in fact what we do in the experiment presented in the next section.

4 Experimental Results

To test the usefulness of expected-frequency interpolation we performed an experiment on the parser described in Section 2. The parser was trained on

[2]The case we have in mind is the probability of a word given we know it is a cardinal number. If it is part of a noun-phrase headed by, say "January" then it is probably a "nice" integer like "31" and thus has a relatively high probability as cardinal numbers go. On the other hand, if it is headed by "%" then there are some integers, like "31," but there are also a lot of very low probability items like "31.65," things we typically do not find modifying "January."

slightly under one million words of the Penn tree-bank Wall Street Journal corpus [5]. It was tested on another 50,000 words of held out testing data. These are all of the sentences in one sub-file of the corpus restricting consideration to those of length less than or equal to 40 words and punctuation. In one trial the λ values were based upon Equation 4 whereas in the second we used Equation 24. In both cases the actual λ values were determined by using the expectation-maximization algorithm to train the λs to optimal values based upon some previously held out training data. The purpose of this is to guarantee that the λ values are the best possible given the general scheme upon which they are based.

As a measure of performance we use what is now the standard measure for this type of parsing, *labeled constituent precision and recall*. A *labeled constituent* is defined by two integers (the begin and end points of the constituent) and by the label of the constituent (e.g., np). A constituent produced by the parser is thus considered correct if there exists a constituent in the correct parse that starts at the same point, ends at the same point, and has the same label.[3] We give both the *precision*, the number of correct constituents produced by the parser divided by the total number of constituents produced, and the *recall*, the number of correct constituents produced by the parser divided by the total number of correct constituents in the "gold standard" (the Penn tree-bank version). We use these measures of performance because (a) they are the standard in this area of parsing, (b) they are closer to measures we care about than, say, cross entropy or perplexity, and (c) it is our observation that measures such as cross entropy and perplexity can vary quite a bit and still have no statistically significant effect on the measures we do care about.

In the following table we give the results of our two runs, along with the best previous results on this exact test, those of Collins [3].

System	Labeled Precision	Labeled Recall
Frequency Interpolation	86.7%	86.4%
Expected-Frequency Interpolation	87.1%	86.8%
Collins	86.3%	85.8%

[3]Actually, to allow comparison to previous work, we use a minor variation of this described in [2,3]. The differences are not relevant to the current discussion.

We see that the expected-frequency system outperforms the one based upon conditioning event counts by .4% in both of our measures.

This may not seem to be a huge difference, but there are several things to keep in mind when judging its significance. The first is that the improvement is essentially for free. The frequence-based system works *identically* to the conditioning event-based system, except it does one extra multiplication specified in Equation 5. Second, we are discussing here a system that even before the performance improvement was already the state of the art (though admittedly, not by much). Given these facts, an improvement of .4% strikes us as quite large indeed.

5 Conclusion

Expected-frequency interpolation is a technique for improving the performance of deleted interpolation smoothing. It allows a system to make finer-grained estimates of how often one would expect to see a particular combination of events than is possible with the traditional frequency interpolation method. This allows the system to better weigh the emphasis given to the various probablity distributions being mixed. This is useful since the more often one would expect things to occur, the more weight should be given to distributions based upon numerous conditioning events, as opposed to probabilities that are less specific, and thus likely to be less accurate. We showed how in some situations the more traditional frequency-interpolation method can lead to anomalous results. We then showed that while the equations for expected-frequency interpolation are not exact, they are close, depending on how well some reasonable assumptions hold. We then presented an experiment in which the introduction of frequence-based interpolation improved performance by .4% with essentially no extra work, and no change in the workings of the system. We also noted that as the system was already the top performer at its task before the change, a .4% improvement was well worth obtaining.

We believe there are three requirements that must be satisfied before expected-frequency interpolation is worth pursuing. First, and most obvious, the assumptions as illustrated in Equations 12 and 21 must be reasonable. This is an important requirement, but one that in most circumstances probably holds well enough. Second, the conditioned event must dramatically affect the expected frequency of the total event package. This is true in our example, where the conditioned event is a particular word. It would be less true if it were, say, the part-of-speech tag for a word, since there are fewer tags by several orders of magnitude. Thirdly, and to our mind most importantly, the probability distribution must be such that different hypotheses (in our case, different parses) condition on quite different events. This is true for parsing, as in our examples where the probabilities are conditioned on quite different head words in different parses. In some other problems this is not the case, most obviously in the case of trigram models for English, where no matter what, we condition on the previous two words in the sentence. The same is true in most tagging models. However, as we get more ambitious, we suspect that more and more probabilistic models will have the more flexible structure that parsing models are now developing. As they do, expected-frequency interpolation should be a good addition to the mix of techniques used therein.

References

1. Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Lai, J. C. and Mercer, R. L. An estimate of an upper bound for the entropy of english. *Computational Linguistics 18*1 (1991).
2. Charniak, E. Statistical parsing with a context-free grammar and lexical information. (Forthcoming, 1996).
3. Collins, M. J. *A new statistical parser based on bigram lexical dependencies.* In *Proceedings of the 34th Annual Meeting of the ACL*. 1996.
4. Hindle, D. and Rooth, M. *Structural ambiguity and lexical relations.* In *Proceedings of the Association for Computational Linguistics*. ACL, 1991, 229 – 236.
5. Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics 19* (1993), 313–330.
6. Merialdo, B. Tagging english text with a probabilistic model. *Computational Linguistics 20*2 (1994), 155–172.