Hiding a Semantic Hierarchy in a Markov Model

Steven Abney
AT&T Labs
180 Park Ave
Florham Park, NJ 07932
abney@research.att.com

Marc Light
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730 USA
light@mitre.org

Abstract

We introduce a new model of selectional preference induction. Unlike previous approaches, we provide a stochastic generation model for the words that appear as arguments of a predicate. More specifically, we define a hidden Markov model with the general shape of a given semantic class hierarchy. This model has a number of attractive features, among them that selectional preference can be seen as distributions over words. Initial results are promising. However, unsupervised parameter estimation has proven problematic. A central problem is word sense ambiguity in the training corpora. We describe attempts to modify the forward-backward algorithm, an EM algorithm, to handle such disambiguation. Although these attempts were unsuccessful at improving performance, we believe they give insight into the nature of the bottlenecks and into the behavior of the EM algorithm.

1 Introduction

We describe here an approach to inducing selectional preferences from text corpora. In the traditional view, a predicate constrains its arguments by selecting for particular semantic classes, or *concepts*. **Selectional restriction** of the traditional sort can be characterized as a relation $\rho(v, r, c)$ over predicates v, syntactic roles r, and argument concepts c. Individual instances (v, r, c) are **selectional tuples**. Examples are given in table 1.

Predicate	Role	Argument Class
splatter	*subj*	CAUSAL-AGENT
splatter	*obj*	FLUID
splatter	*on*	SURFACE

Table 1: Selectional tuples

Of more interest to computational linguistics is **selectional preference**, a continuous-valued generalization of selectional restriction. Selectional preference is a mapping $\sigma : (v, r, c) \mapsto a$ that maps each tuple (v, r, c) to a real number a, the **degree of preference** of v for c with respect to role r. Positive degrees of preference are intended to correlate with intuitive judgments of "plausibility" or "typicality," and negative judgments are intended to correlate with intuitive judgments of "implausibility."

We have chosen to characterize such selectional preference as a side-effect of a stochastic model for generating what we will call **co-occurrence tuples**: triples (v, r, n) for v a predicate, r a syntactic role, and n the headword of the argument filling the role r with respect to v. An example of a co-occurrence tuple is (*splatter*, obj, *water*). Co-occurrence tuples can be obtained from text corpora, and can be used to make inferences about the probability of selectional tuples. For example, the co-occurrence tuple (*splatter*, obj, *water*) may be taken as evidence for the selectional tuple (*splatter*, obj, FLUID). More concretely, such co-occurrence tuples make up the training corpora, from which we train our stochastic models.

For this study, we have used the British National Corpus (100M words), from which we have extracted co-occurrence tuples using the Cass parser (Abney, 1997). By way of illustration, table 2 shows the values of n in tuples (*eat*, obj, n) along with their frequencies in the corpus. This "subcorpus" would be used to train a stochastic model specific to the object role of the verb *eat* and is the first of two inputs to our induction process.

There are two problems with such training data: it is noisy and it contains ambiguity. The noise is sometimes due to tagging or parsing errors, and sometimes due to metaphorical uses. Examples from

meat	45	bucket	1	ice	2
tape	1	investment	1	soup	2
proportion	2	kitchen	1	fry	4
root	4	salad	2	top	1
bread	14	feast	1	scrap	2
majority	2	sauce	1	sugar	1
principle	1	food	77	hole	2
roll	4	pack	1	bag	2
race	1	mouthful	3	dinner	11
sheep	1	salt	1	meal	46
trout	2	pasta	1	slice	7
dish	2	spaghetti	6	chicken	5
stick	1	egg	18	average	1
sandwich	13	yogurt	1	mustard	1
breakfast	30	garlic	1		

Table 2: Objects of *eat* in the BNC

table 2 include *investment, average, tape*, and *race*. However, note that the "good" examples such as *food* and *meal* are much greater in number and frequency. Thus, the signal is stronger than the noise in most cases and most reasonably robust training methods will be able to handle the noise.

The second problem, that of word sense ambiguity, is more difficult. The word *bread* in table 2 provides an example. *Bread* can be used to refer to a food, e.g., *the multigrain bread in Germany is wonderful*, but it can also refer to money, e.g., *I could really use some bread since my car just broke down.* For this reason, it is not immediately clear which concepts the 14 tokens of *bread* provide evidence for. If the wrong choice is made for a high frequency word, incorrect selectional preferences will result.

The model we propose represents this sort of uncertainty in a natural way: the two senses of *bread* are represented as different paths through a stochastic model, both of which generate the same observation. This stochastic model is a hidden Markov model (HMM) which has the shape of a given semantic hierarchy. Figure 1 shows an example hierarchy. In the work discussed here, we made use of the WordNet semantic hierarchy (Miller, 1990). This hierarchy is the second input to our induction process.

We hoped that the forward-backward algorithm, an EM algorithm, would properly disambiguate word senses in the training data as a side effect of its quest to maximize the likelihood of the training data given the model. However, for reasons we will discuss in section 4, this was not the case.

In the following section we discuss work on selectional preference induction that also assumes as input (i) subcorpora corresponding to predicate role pair and (ii) a semantic class hierarchy. Then we

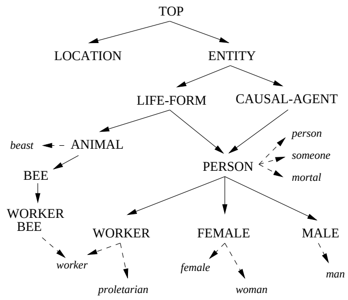

Figure 1: Example Semantic Class Hierarchy

formally define our stochastic model. Next we look at a number of ultimately unsuccessful attempts to modify the forward-backward algorithm to perform effective word-sense disambiguation of the training data. Despite these problems we did obtain some encouraging results which we present at the end of the paper.

2 Related work

There have been a number of attempts to derive selectional preferences using parsed corpora and a semantic class hierarchy. Our work is closely related to that of (Resnik, 1993). His system provides a distribution over classes conditioned on a predicate-role pair: $p(c|v,r)$. It estimates $p(c|v,r)$ as $f(v,r,c)/\sum_{c'} f(v,r,c')$, where $f(v,r,c)$ is in turn approximated by allocating the frequency of the co-occurrence tuple (v,r,n) among the classes $C(n)$ to which the senses of n belong. For example, suppose the word *bread* has two senses, BREAD and MONEY. Suppose further that BREAD is a hyponym of BAKED-GOODS, FOOD, ARTIFACT, and TOP, and MONEY is a hyponym solely of TOP. Then $C(\textit{bread})$ is {BREAD, BAKED-GOODS, FOOD, ARTIFACT, TOP, MONEY}. Tokens of *bread* are taken as ambiguous evidence for all concepts in $C(\textit{bread})$; the weight of evidence is divided uniformly across $C(\textit{bread})$. Hence each token of (*eat*, obj, *bread*) counts as 1/6 of a token of (*eat*, obj, BREAD), 1/6 of a token of (*eat*, obj, BAKED-GOODS), and so on. Such a uniform allotment is does not reflect empirical distributions of senses, which are Zipf-like, but does produce reasonable results. It is important to note that Resnik is not very explicit about how the probability $p(c|v,r)$ is to be interpreted; there is no explicit stochastic generation model involved.

Resnik uses $p(c|v,r)$ to quantify selectional preference by comparing it to $p(c)$, the marginal probability of class c appearing as an argument. He measures the difference between these distributions as their relative entropy (D). The total amount of "selection" that a predicate v imposes on the filler of role r is quantified as $D(p(c|v,r)||p(c))$. The selectional preference of v for c in role r is quantified as the contribution of the c to the total amount of selection:

$$selpref(v, r, c) = \frac{p(c|v,r) \log \frac{p(c|v,r)}{p(c)}}{D(p(c'|v,r)||p(c'))}$$

The class or classes produced as the output for the predicate are those with the highest *selpref* value.

Other work on the induction of selectional preferences includes (Li and Abe, 1995). They characterize the selectional restriction of a predicate with a horizontal cut through a semantic hierarchy, and use the principle of Minimum Description Length (MDL) to choose a cut that optimally balances simplicity and descriptive adequacy. More specifically, a cut is a set of concepts that partition the set of nouns belonging to the hierarchy. A cut is deemed simpler if it cuts the hierarchy at a higher place (i.e., the cut contains fewer concepts), and descriptive adequacy is measured by comparing the actual distribution of nouns filling a slot (v, r) to the closest approximation one can obtain by estimating $p(n|c)$ for only the concepts c in the cut. Again, the intended stochastic generation model is not clear.

As mentioned, the interpretation of expressions such as $p(c|v,r)$ is obscure in these previous models. Without clarity about what stochastic process is producing the data, it is difficult to gauge how well probabilities are being estimated. In addition, having an explicit stochastic generation model enables one to do a number of things. First, one can experiment with different methods of eliminating word sense ambiguity in the training corpus in a principled fashion. Second, it is often possible to calculate a number of useful distributions. From our model, the following distributions can be efficiently estimated: $Pr(word|predicate, role)$, $Pr(word|semantic\text{-}class, predicate, role)$, and $Pr(word\text{-}sense|word, predicate, role)$. These distributions can be used directly to help solve ambiguity resolution problems such as syntactic structure disambiguation. In addition, the $Pr(word|predicate, role)$ distribution can be seen as a very specific language model, i.e., a language model for the head of the argument of the predicate.

3 Our Stochastic Generation Model

Our model generates co-occurrence tuples (e.g., (*eat*, obj, *beef*)) as follows. The probability $p(v, r, n)$ of a co-occurrence tuple can be expressed as $p(v,r)p(n|v,r)$. Our central concern is the conditional probability $p(n|v,r)$. We associate a separate HMM with each pair (v, r) in order to characterize the distribution $p(n|v,r)$. Thus, the HMM for (*eat*, obj) would be different than that for (*drink*, subj). That is, the general structure of the HMM would be the same but the parameters would be different.

The states and transitions of the HMMs are identified with the nodes and arcs of a given semantic class hierarchy. The nodes of the hierarchy represent semantic classes (concepts), and the arcs represent hyponymy (that is, the "is-a" relation). Some concepts are expressible as words: these concepts are **word senses**. A sense may be expressible by multiple words (synonyms) and, conversely, a single word may be an expression of more than one sense (word sense ambiguity). For expository reasons, we assume that all and only the terminal nodes of the hierarchy are word senses. In actuality, the only constraint our system places on the shape of the hierarchy is that it have a single root.

A "run" of one of our HMMs begins at the root of the semantic hierarchy. A child concept is chosen in accordance with the HMM's transition probabilities. This is done repeatedly until a terminal node (word sense) c is reached, at which point a word w is emitted in accordance with the probability of expressing sense c as word w. Hence, each HMM "run" can be identified with a path through the hierarchy from the root to a word sense, plus the word that was generated from the word sense. Also, every observation sequence generated by our HMMs consists of a single noun: each run leads to a final state, at which point exactly one word is emitted.

More formally, a concept graph is given, and an expressibility relation from nodes to words. The nodes of the graph are identified with concepts $\mathcal{C} = \{c_1, \ldots, c_n\}$, and the expressibility relation relates concepts to words $\mathcal{W} = \{w_1, \ldots, w_m\}$. The HMM consists of a set of **states** $\{q_1, \ldots, q_n\}$, which we identify with the nodes of the concept graph; a set of **possible emissions** which we identify with $\mathcal{W} \cup \{\epsilon\}$ (that is, we permit non-emitting states); and three parameter matrices:

$A = \{a_{ij}\}$ The transition probabilities. The value a_{ij} represents the probability of making a transition from state q_i to state q_j. a_{ij} is nonzero only if there is an arc in the concept graph from concept c_i to concept c_j.

$B = \{b_j(k)\}$ The emission probabilities. The value $b_j(k)$ represents the probability of emitting word w_k while in state q_j. States corresponding to nonterminal nodes in the concept graph are non-emitting (that is, they emit ϵ with probability 1), and states corresponding to terminal nodes are emitting states (they emit ϵ with probability 0).

$\pi = \{\pi_i\}$ The initial state distribution. π_i is identically 1 for the start state (corresponding to the root node), and 0 for all other states.

As mentioned, we associate an HMM with each pair (v, r). Each HMM has the same structure, determined by the semantic hierarchy. Where they differ is in the values of the associated parameters. To estimate parameters, we require a training sample of observation sequences. Since each observation sequence consists of a single word, a training sample is simply a collection of word tokens. The training sample consists of the nouns filling the associated "slot" (v, r)—that is, a token of the noun n is included in the training sample for each token of the tuple (v, r, n) that occurs in the corpus. Table 2 provides an example corpus.

This approach permits us to address both word sense disambiguation and selectional preference. An ambiguous word is one that could have been generated by means of more than one state sequence. For a given ambiguous word n appearing in a slot (v, r), we can readily compute the posterior probability that word sense c was used to generate n, according to the (v, r) model. We can disambiguate by choosing the word sense with maximum posterior probability, or we can use the probabilities in a more sophisticated model that uses more contextual information than just the slot in which the word appears.

Selectional preferences for (v, r), can be extracted from these models by calculating the distribution over classes $p(c|v, r)$ from the model trained for (v, r) and the distribution $p(c)$ from a model trained on all nouns. One can then follow Resnik and use *selpref*(v,r,c) as defined above. These distributions can be calculated by considering our HMMs with additional transitions going from all leaf states to the root state. Such HMMs are ergodic and thus the probability of being in a particular state at a time t converges to a single value as t approaches ∞. These steady-state probabilities can be put entirely in terms of the parameters of the model. Thus, once an HMM has been trained, the steady state probabilities can be easily calculated. Because of the correspondence between states and classes, these steady state distributions can be interpreted as a distribution over classes.

As mentioned earlier, another way of thinking about selectional preference is as a distribution over words. For example, the selectional preference of the verb *eat* for its direct object would be expressed by high probabilities for words like *breakfast, meat,* and *bread* and low probabilities for words like *thought, computer,* and *break.* This conception of selectional preference is related to language modeling in speech recognition. In fact, the selectional preference of a predicate-role pair can be thought of as a very specific language model. This way of thinking about selectional preferences is useful because it points to possible applications in speech recognition.

4 Parameter Estimation

We had originally hoped that after turning our semantic hierarchy into an HMM as described above, we could simply run the standard forward-backward algorithm on the training corpus and we would get a useful model. Unfortunately, there are a number of reasons why this does not work. We will describe these problems and our attempted solutions in the context of disambiguating the words in the training data with multiple word senses, a fundamental task in the estimation of selectional preferences. In each of the three sub-sections below we describe a problem we discovered and an attempted solution. In the end, we were not able to produce a system that performed better than Resnik's system on his word-sense disambiguation evaluation. This evaluation is an indirect way of testing whether the training method is word sense disambiguating the training corpora correctly. However, when we derived from our models a ranked list of classes using $p(c|v, r)$ and Divergence as described above, we obtained very good lists. We present some representative lists and the results on Resnik's evaluation in section 5.

In addition, we think the attempted solutions are instructive and provide insight into the nature of the problem and the behavior of the EM algorithm.

4.1 Smoothing

It was our original hope that, by treating the choice of word sense as just another hidden variable in the HMM, word-sense disambiguation would be accomplished as a side effect of EM estimation. In fact, however, there is no pressure in the model in favor of parameter settings in which occurrences of an ambiguous word are all accounted for by a single word sense. If the initial parameter settings account for an ambiguous word as a mixture of word senses, the converged model does likewise. This should come as no surprise to those with experience using EM, but is not usually stated very clearly in the literature: the EM algorithm estimates a mixture model and (intuitively speaking) strongly prefers mixtures containing small amounts of many solutions over mixtures that are dominated by any one solution.

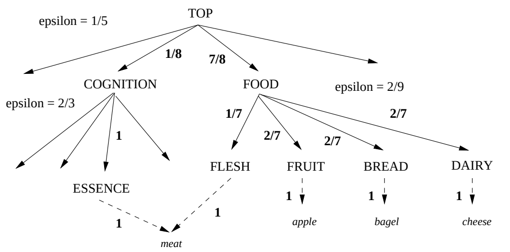

Figure 2: Smoothing

For example, consider Figure 2. We assume a miniature training corpus, containing one instance each of four words, *meat, apple, bagel, cheese.* The word *meat* is ambiguous, having both sense ESSENCE and sense FLESH. The training corpus is perfectly accounted for by the weights in Figure 2, and this is indeed a fixed point of the EM algorithm.

One would like to introduce some pressure toward consolidating word occurrences under a single word sense. Further, one would like the set of word senses one ends up with to be as closely related as possible. In Figure 2, for example, one would like word *meat* to shift as much of its weight as possible to sense FLESH, not sense the ESSENCE.

We sought to accomplish this in a natural way by smoothing transition probabilities, as follows. The transition probabilities out of a given state constitute a probability distribution. At a given iteration of the EM algorithm, the "empirical" distribution for a given state is the distribution of counts across outgoing transitions, where the counts are estimated using the model produced by the previous iteration. (Hence the scare quotes around *empirical.* For want of a better term, let us call this distribution *pseudo-empirical.*)

For example, assume the parameter settings shown in Figure 2 to be the output of the previous iteration, and assume that each word appears once in the training corpus. Then the (estimated) count for the path through transition FOOD $\rightarrow$ FLESH is $1/2$, and the count for the paths through transitions FOOD $\rightarrow$ FRUIT, FOOD $\rightarrow$ BREAD, FOOD $\rightarrow$ DAIRY is 1 each. Hence, the total count for the state FOOD is 3.5. Dividing each transition count by the count for state FOOD yields the pseudo-empirical probabilities $\{1/7, 2/7, 2/7, 2/7\}$.

The pseudo-empirical probabilities would normally be installed as transition weights in the new model. Instead, we mix them with the uniform distribution $\{1/4, 1/4, 1/4, 1/4\}$. Let $p(t)$ be the pseudo-empirical probability of transition t, and let $u(t)$ be the uniform probability of transition t. Instead of setting the new weight for t to $p(t)$, we set it to $\varepsilon\, u(t) + (1-\varepsilon)p(t)$.

Crucially, we make the mixing parameter, ε, a function of the total count for the state. Intuitively, if there is a lot of empirical evidence for the distribution, we rely on it, and if there is not much empirical evidence, we mix in a larger proportion of the uniform distribution. To be precise, we compute ε as $1/(c+1)$, for c the total count of the state. This has the desirable property that ε is 1 when c is 0, and ε decreases exponentially with increasing c.

It is probably not immediately obvious how smoothing in this manner helps to prune undesired word senses. To explain, consider what happens in Figure 2. There are two paths from the root to the word *meat*, one leading through the word sense ESSENCE and the other leading through the word sense FLESH. In the "previous" model (i.e., the weights shown), each of those paths has the same weight (namely, $1/8$), hence each instance of the word *meat* in the training corpus is taken as evidence in equal parts for word senses ESSENCE and FLESH.

The difference lies in the states COGNITION and FOOD. Words *apple, bagel,* and *cheese*, along with half of *meat*, provide evidence for the state FOOD, giving it a total count of $31/2$; but the only evidence for state COGNITION is the other half of *meat*, giving it a total count of $1/2$. The new distribution for COGNITION has a large admixture of the uniform distribution, whereas the distribution of FOOD has a much smaller uniform component.

The large proportion of uniform probability for the state COGNITION causes much of its probability mass to be "bled off" onto siblings of ESSENCE (not shown, but indicated by the additional outgoing edges from COGNITION). Since none of these sibling are attested in the training corpus, this makes COGNITION's fit to the training corpus very poor. Intuitively, this creates pressure for TOP to reduce the weight it apportions to COGNITION and increase its weight for FOOD; doing so improves the model's overall fit to the training corpus.

This decreases the relative count for the word sense ESSENCE in the next iteration, increasing the pressure to shift weight from COGNITION to FOOD. Ultimately, an equilibrium is reached in which most of the count for word *meat* is assigned to the word sense FLESH. (What prevents a total shift to the word sense FLESH is smoothing at TOP , which keeps a small amount of weight on COGNITION. In a large hierarchy, this translates to a vanishingly small amount of weight on ESSENCE.)

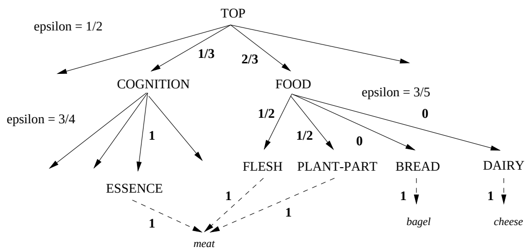

Figure 3: Imbalanced Senses

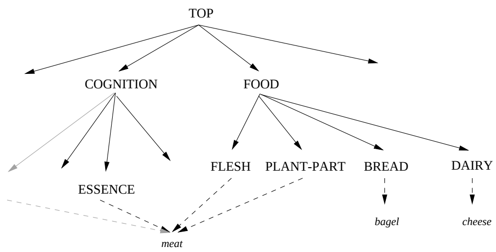

Figure 4: Sense Balancing

4.2 Sense Balancing

In Figure 2, our smoothing method produces the desired bias for the corpus *meat, apple, bagel, cheese.* However, in different circumstances the bias produced is not the desired one. Consider training the hierarchy in Figure 3 on a corpus made up of one token of *meat*.

The hierarchy in Figure 3 differs from the hierarchy in Figure 2 in that *meat* has three senses, two of which share a prefix path, i.e., the transition from TOP to FOOD. When training on the corpus of one token of *meat*, $2/3$ of the count would go down the FOOD side and the other third down the COGNITION side; thus, with respect to the forward-backward algorithm, there is little difference between the current example and the previous one. Therefore, the two senses of *meat* under FOOD will be preferred. Intuitively this is wrong, because there is no information in the corpus on which to derive a bias for any one sense and we would like our parameter settings to reflect this. In addition, this is also not simply a border case problem, since if *meat* is very frequent, as in the corpus in Table 2, it could easily happen that such an a priori bias for certain senses of *meat* drowns out the bias that should result from the other words in the corpus.

In concrete terms, the problem is the shared path prefix that exists for the senses under FOOD, namely the transition from TOP to FOOD. More abstractly, the problem is that the hierarchy is not balanced with respect to the senses of *meat*—if there were another sense under ESSENCE there would be no problem (see Figure 4).

One can simulate such a phantom sense within the forward-backward algorithm. First the count for the transitions in the prefix path have to be reduced. This can be done by modifying the E step such that the expectation, $\widehat{E_w}(X_{i\to j})$, for the random variable, $X_{i\to j}$, which corresponds to the transition from state i to state j for a single token of word w, is calculated

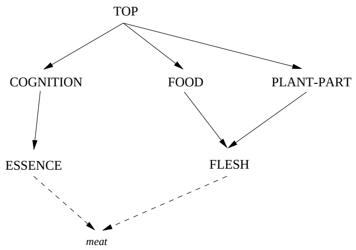

Figure 5: Graph with Reentrancy

as follows.

$$\widehat{E_w}(X_{i\to j}) = \frac{E_w(X_{i\to j})}{\mathcal{D}(j,w)}$$

where $E_w()$ is the expectation based on the model and corpus and $\mathcal{D}(j,w)$ is the number of unique paths starting at j and ending in a state that can generate w. One then sums over all tokens of the corpus to get the expectation for the corpus.

The second step is to reduce the probability of the paths to the sister sense of the phantom sense, e.g., COGNITION$\to$ ESSENCE. This can be achieved by increasing the normalization factor used in the M step:

$$\widehat{A_w} = \widehat{E_w}(\mathcal{D}(r,w) - \mathcal{D}(i,w))$$

Once again, we focus on the contribution of a single token of a word w and thus the normalization factor used in the M step would be the sum $\widehat{A_w}$ over the tokens in the corpus. The state r is the starting state of the model, i.e., the state corresponding to the root of the hierarchy. The exception to this formula occurs when $\mathcal{D}(r,w) - \mathcal{D}(i,w) = 0$, in which case $\widehat{A_w} = \widehat{E_w}$.

There are other ways of modifying the algorithm to simulate the phantom sense. However, this method is easy and efficient to implement since the E and M steps remain simple local calculations—the only global information comes through the function d which can be efficiently and easily computed.

Another kind of sense imbalance is shown in Figure 5. This imbalance can be corrected by further

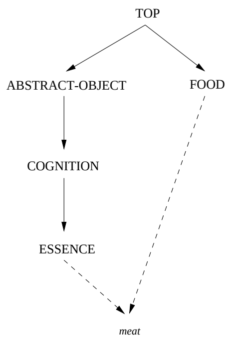

Figure 6: Path on the Right is Preferred Due to its Shorter Length

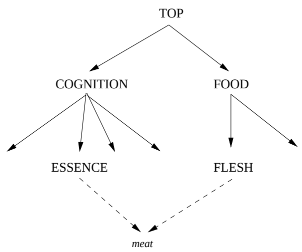

Figure 7: Path on the Right is Preferred Due to its Relatively Narrow Distributions

modifying the E step as follows:

$$\widehat{E_w}(X_{i \to j}) = \frac{E_w(X_{i \to j})}{\mathcal{D}(j, w)\mathcal{U}(j)}$$

where $\mathcal{U}(j)$ is the number of unique paths up to the root from j.

4.3 Length and Width Balancing

Most of the example hierarchies/models we have considered so far have been balanced with respect to length and width, i.e., the length of the paths to the generating states has been uniform and the number of transitions out of a state has been uniform across states. It turns out that uniform length and width are important characteristics with respect to our modified forward-backward algorithm: shorter paths are preferred to longer ones (see Figure 6) and paths that go through states with few exiting transitions are preferred to ones that go through states with many (see Figure 7). In fact, short paths are preferred to longer ones by the standard forward-backward algorithm, since in an HMM the probabilities of events in a sequence are multiplied to get the probability of the sequence as a whole. Width only comes into play when one introduces smoothing. Remember that in our smoothing, we mix in the uniform probability. Consider the transitions coming out of the state COGNITION in Figure 7; there are four transitions and thus the uniform probability would be 1/4. In contrast, the transitions coming out of the state FOOD in the same figure number only 2 and thus the uniform distribution would be 1/2. If there are many transitions the probability mixed for the uniform distribution will be smaller than if there were fewer transitions.

We can solve the problem by balancing the hierarchy: all paths that result in generating a symbol should be of the same length and all distributions should contain the same number of members. As in the previous section, we can simulate this balancing by modifying the forward-backward algorithm.

First, to balance for width, the smoothing can be modified as follows: instead of mixing in the uniform probability for a particular parameter, always mix in the same probability, namely the uniform probability of the largest distribution, u_{max} (i.e., the state with the largest number of exiting transitions; in Figure 7, this maximum uniform probability would be 1/4). Thus the smoothing formula becomes $\varepsilon\, u_{max} + (1-\varepsilon)p(t)$. This modification has the following effect: it is as if there are always the same number of transitions out of a class. Width balancing for emission parameters is performed in an analogous fashion.

Let us turn to length balancing. Conceptually, in order to balance for length, extra transitions and states need to be added to short paths so that they are as long as the maximum length path of the hierarchy. It should be noted that we are only concerned with paths that end in a state that generates words. The extension of short paths can be simulated by multiplying the probability of a path by a factor that is dependent on its length:

$$\widehat{Prob}(p) = Prob(p) {u_{max}}^{(length_{max} - length(p))}$$

This additional factor can be worked into the forward and backward variable calculations so that there is no loss in efficiency. It is, thus, as if $length_{max} - length(p)$ states have been added and that each of these states has ${u_{max}}^{-1}$ exiting transitions.

5 Preliminary Results

As mentioned above, we tested our trained models on a word-sense disambiguation evaluation, reasoning that if it performed poorly on this evaluation, then it must not be disambiguating the training corpus very well. The bottom line is that we were not able to advance the state of the art—the performance results are comparable to, but not better than, those obtained by Resnik. We used the training sets, test sets, and evaluation method described in (Resnik, 1997).[1] Table 3 presents performance results. The Random method is simply to randomly pick a sense with a uniform distribution. The First Sense method is to always pick the most common sense as listed in WordNet. The HMM smoothed method is to use models trained with smoothing but no balancing modifications. HMM balanced uses smoothing and all three balancing modifications.

Method	
Random	28.5%
First Sense	82.8%
Resnik	44.3%
HMM smoothed	35.6%
HMM balanced	42.3%

Table 3: Word Sense Disambiguation Results

Next we give examples of the preferences derived from trained models for three verbs, , represented as weights on classes. These are typical rather than best-case examples. We have not yet attempted any formal evaluation of these lists.

eat	0.321048	food
	0.245948	substance
	0.209142	nutriment
	0.156176	object
	0.144745	entity
	0.072242	meal
abandon	0.078877	content
	0.061569	psychological feature
	0.057775	idea
	0.056840	cognition
	0.038888	plan
	0.025118	activity
	0.023805	attempt
	0.023058	act
	0.021834	belief
break	0.033223	object
	0.020298	law
	0.020287	law of nature
	0.018689	substance
	0.016658	ice
	0.016407	solid
	0.015359	guidance
	0.014416	rule
	0.014345	entity
	0.014334	crystal

[1]We would like to thank Philip Resnik for providing us with the training and test data that he used in the above mentioned work.

6 Conclusion

In the last section, we showed why the straightforward application of an EM algorithm, namely the forward-backward algorithm, would not disambiguate the sensese of input words as desired. Thus, we introduced a type of smoothing which produced the desired bias in the example at hand. Then we showed how this smoothing, when used on certain graphs, produced unwanted biases which then necessitated further modifications in the E and M steps of the algorithm. In the end, even with smoothing, sense, length, and width balancing, the performance of the EM-like estimation was disappointing.

One possible lesson is that EM itself is inappropriate for this problem. Despite the fact that it has become the default method for uncovering hidden structure in NLP problems, it essentially averages together many possible solutions. Possibly, a less linear method that eventually commits to one or another hypothesis about hidden structure may be more appropriate in this case.

In conclusion, this paper has made the following contributions: it has shown how a stochastic generation model can make use of a semantic class hierarchy, it has provided a negative result with respect to parameter estimation for this model, and in doing so has provided an interesting illustration of the inner workings of the forward-backward algorithm.

References

Abney, Steven 1997. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4).

Li, Hang and Abe, Naoki 1995. Generalizing case frames using a thesaurus and the MDL principle. In *Proceedings of Recent Advances in Natural Language Processing*.

Miller, George 1990. WordNet: An On-Line Lexical Database. *International Journal of Lexicography*, 3(4).

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

Resnik, Philip *Selection and Information: A Class-Based Approach to Lexical Relationships.* PhD thesis, University of Pennsylvania, Philadelphia, PN, 1993.

Resnik, Philip Selectional preference and sense disambiguation. In *Proceedings of the ANLP-97 Workshop: Tagging Text with Lexical Semantics: Why, What, and How?*, Washington, D.C., 1997.