Statistical Language Modeling Using Grammatical Information

Dennis Grinberg

September 11, 1995

Abstract

We propose to investigate the use of grammatical information to build improved statistical language models. Until recently, language models were primarily influenced by local lexical constraints. Today, language models often utilize longer range lexical information to aid in their predictions. All of these language models ignore grammatical considerations other than those induced by the statistics of lexical constraints. We believe that properly incorporating additional grammatical structure will achieve improved language models.

We will use link grammar as our grammatical base. Being highly lexical in nature, the link grammar formalism will allow us to integrate more traditional modeling schemes with grammatical ones. An efficient robust link grammar parser will assist in this undertaking.

We will initially build finite state-based language models that will utilize relatively simple grammatical information, such as part-of-speech data, along with information sources used by other language models. Our models feature a new framework for probabilistic automata that makes use of hidden data to construct context-sensitive probabilities. The maximum entropy principle employed by these *Gibbs-Markov models* facilitates the easy integration of multiple information sources.

We will also build language models that take greater advantage of link grammar by including more sophisticated grammatical considerations. These models will include both probabilistic automata as well as models more closely related to the link grammar formalism.

Expected contributions of this work will be to demonstrate that grammatical information can be used to construct language models with low perplexity, and that such models can be used to reduce the error rates of speech recognition systems.

1. Introduction

In recent years speech recognition systems have become more prevalent in our society. Many commercial systems impose numerous restrictions in order to achieve adequate recognition accuracy, while at the same time the speech recognition research community works to remove these restrictions. Improving language models is one area of research towards this goal of removing such restrictions.

Language modeling is the attempt to estimate the probabilities of occurrences of word sequences in a natural language. The probabilities are determined by the values of various parameters of the language model. The parameters are computed by *training* the model on large quantities of data in the language. These parameters correspond to various features of the language as determined by the statistics of the training data. The most common information used to train language models is gathered from frequency counts of short sequences of words, called *n-grams*, in the training data. This information, though extremely limited, still allows language models to perform surprisingly well. But in order to further improve speech recognition systems, it appears that language models will need to include more sophisticated knowledge about the language.

Intuitively, it would seem that grammatical syntax should play an important role in a language model's predictions. Syntax defines when a sentence is well-formed. By examining the syntactic roles played by lexical items in sentences, important semantic restrictions of the natural language can be inferred. In spite of the potential usefulness of syntactic structure, improvements to language models have not come from this direction. While there have been attempts to incorporate syntactic structure in language models, these attempts have not achieved the improvements we would hope to achieve when using such a rich information source. We believe that most of these attempts were unsuccessful because they were overly concerned with syntax and neglected the importance of the lexical items themselves. We believe that language models can be improved by utilizing syntactic structure if they do not lose sight of the semantic restrictions imposed by relations among the words themselves.

We propose using link grammar as an information source to aid in building improved language models. While equivalent in expressive power to context-free grammars, link grammars are defined in terms of syntactic relations among words. This makes it easier to model the semantic restrictions imposed by both local relations among words and the syntactic roles in which they participate.

In order to fully utilize link grammar, we need to be able to parse text from the domain we intend to model. While it is difficult to "parse" unrestricted text, we will be assisted by a robust link grammar parser. This parser will efficiently extract useful structural information from sentences, even when they are highly ungrammatical.

Our work will involve the integration of traditional information sources, such as n-grams and triggers, with grammatical information extracted using the robust link grammar parser on a corpus of spontaneous speech. This information will be used to train language models in a *Gibbs-Markov* framework. Gibbs-Markov models may be viewed as extensions of hidden Markov models, allowing more general interactions among the hidden states and output symbols. These models are further categorized by their parameterization as exponential models, facilitating their efficient training in a maximum entropy framework. We hope to incorporate our language models into a speech recognition system.

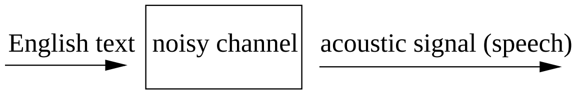


Figure 1: English is passed through a channel that produces an acoustic signal

2. Language modeling

Language modeling attempts to capture important properties of a natural language by exploiting statistics gathered from analyzing corpora in the language. Differences in language models depend most significantly on which features of the language are modeled. In the next few subsections we will more precisely describe the role of language models and explain how data is used to train them. We will also describe a number of popular language models.

2.1. Description of language modeling

Language models model a natural language by assigning probabilities to strings of words in the language. These probabilities can then be used to rank word sequences. A good language model will provide rankings of sentences that correspond with our intuition (or other criteria) of what likely sentences are. Language modeling is used in numerous applications including speech recognition, handwriting recognition, optical character recognition, machine translation, and language generation. We will focus on the application of speech recognition in this work but most of the results are relevant to the other applications as well. In addition, we will concentrate on English as our natural language.

The speech recognition problem can be cast in terms of the noisy channel model from information theory. As shown in Figure 1, English is transmitted over a noisy communications channel that results in an acoustic signal corresponding to speech. Given this acoustic signal, the speech recognition problem is to find the original English that produced the signal. The information theoretic approach searches for the word sequence $\hat{W}$ having the largest posterior probability $P(\hat{W}|A)$. Now

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)}, \quad \text{so}$$

$$\hat{W} = \text{argmax}_W P(A|W)P(W).$$

The conditional probabilities $P(A|W)$ form the *acoustic model.* The probabilities $P(W)$ comprise the *language model*, and are interpreted as the prior probability that W was supplied to the channel. The objective of language modeling is to estimate this prior distribution.

This description of language modeling was given in terms of a complete natural language. But because natural languages are vast, it is extremely difficult to assign meaningful probabilities to their sentences. Therefore, we can limit language models to provide good estimates of $P(W)$ for sub-languages by restricting such factors as the vocabulary of the language, the general topic of discourse (such as "financial news-wire") and the type of language (such as spoken, written, etc.). For many applications, these restrictions should not be conceived of as limitations since they are actually quite beneficial. Many speech recognition systems, by design, have expectations about the variety of English they will need to recognize. These restrictions allow these systems to be more compact, run faster, and to perform more accurately.

2.2. Histories

For a word sequence $W = w_1 w_2 \cdots w_N$ the joint probability $P(W)$ can be decomposed as a product of conditional probabilities:

$$P(W) \quad = \quad \prod_{i=1}^{N} P(w_i | w_1, w_2, \ldots, w_{i-1}). \tag{1}$$

In the term $P(w_i|w_1, w_2, \ldots, w_{i-1})$, $\{w_1, w_2, \ldots, w_{i-1}\}$ is known as w_i's *history*. This formulation allows left-to-right processing of the input text allowing language modeling to be viewed as trying to predict the next word given the previous words in the text. This view of language modeling is quite convenient and we will use it extensively.

How can we calculate the probabilities used in equation (1)? Typically, a *training corpus* (or *training text*) written in the sub-language being modeled is used to derive approximations to these probabilities. Knowing the relative frequency of the appearance of a word with given history in the corpus gives us an indication of the probability to assign that event. The problem with this approach is that more often then not, when using the language model, most histories encountered will either not have appeared in the training text at all or will have appeared too few times to infer any meaningful statistics about them. Even if we limit the size of histories by beginning them anew at the beginning of each sentence, this *sparse data* problem would still be acute.

Our approach to this problem is to group histories together into equivalence classes. The equivalence classes can be chosen deterministically as in *n-gram models*. In these models a class is completely determined by the $n-1$ most recent words in the history, with all histories ending in the same $n-1$ words being grouped together. The classes can also be chosen probabilistically as in *part-of-speech based models*. One such model might assign a probability to the part-of-speech of the current word based on a portion of the history and then assign a probability to the current word based on its part-of-speech. In this case, the terms in equation (1) are expressed as

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \sum_s P(w_i|s) P(s|w_{i-n+1}, \ldots, w_{i-1}).$$

One of the ways in which language models differ from one another is in the equivalence classes they use. The equivalence classes must be specific enough to allow for good predictions of the current word yet they must be general enough so that the sparse data problem won't be acute. In the *Shannon Game*, humans predict unknown words (or letters) from past history. Trials of this game indicate that a number of different aspects of the history such as topic, tense, number, and structure were found to be used with both syntactic and semantic considerations playing important roles in the humans' predictions. Shannon [43] reported results for a variation of this game where humans predicted letters using an alphabet of 27 symbols (26 letters and the space). The result was an upper-bound estimate of the entropy of English of 1.30 bits per character. In [7], a language model was used to estimate the entropy of mixed-case English and an upper bound of 1.75 bits per character was reported. Their case-component contributed 0.04 bits to this figure, so the actual difference in entropy between the human prediction reported by Shannon and the language model prediction is 0.41 bits per character. We conclude that humans are much better at the task of prediction given history than today's language models.

2.3. Review of some language models

In this section we will review some popular language models. We will be begin with the popular n-gram models which still remain the backbones of most systems. We will then discuss class-based models in which the number of parameters can be significantly decreased over n-gram models at the expense of only slightly higher perplexity. There is a general framework in which many language models can be placed. These hidden Markov models are described next. Finally, trigger models, which have recently been shown to have great potential, are described.

2.3.1. n-gram models

One of the most popular types of language models are n-gram models. In these models, the histories are partitioned into classes that depend on the most recent $n-1$ words [1]. The most common n-gram models are *trigram* models in which $n = 3$. For n-gram models, maximum likelihood estimates of the conditional probabilities in (1) are obtained as simple relative frequencies:

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = P(w_i|w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1} \cdots w_i)}{\sum_w C(w_{i-n+1} \cdots w_{i-1} w)}$$

where $C(\cdot)$ counts the number of occurrences of a string in the training text. This method captures local semantic and syntactic constraints in the text by sheer force of numbers. For example, when discussing air pollution, the three word sequence, or *trigram* "smoke from steel" makes much more sense than the trigram "smoke from steal." A trigram language model trained on a corpus relating to air pollution would prefer the former trigram as it would appear more often in the training corpus.

n-grams themselves are prone to the sparse data problem. The probability of observing an unknown n-gram in previously unseen data can be estimated by the Good-Turing estimate [16] as n_1/N where n_1 is the number of n-grams that appeared only once in a sample corpus and N is the size of the corpus. This sparse data problem can be quite acute. For example, Jelinek states in [19] that for a training corpus of patent descriptions whose size was 1,500,000 words, when tested against 300,000 words of previously unseen data from the same corpus, 23% of the trigrams in the unseen corpus had never appeared in the training corpus. In general, as a solution, trigram models are often *smoothed* with 2-gram (*bigram*) and 1-gram (*unigram*) models by taking a linear combination of all three models to provide more accurate estimates for trigrams that either didn't appear or only infrequently appeared in the training text.

One of the limitations of n-gram models is that the only information in the histories being used is local information appearing immediately prior to the word being predicted. Often other parts of the history should affect a word's likelihood of appearing. For example, predicting the sentence "The air pollution index should decrease when the new emissions standards go into effect" would be easier with longer-range dependencies: the chance of "emissions" appearing should be positively affected by "pollution" appearing earlier. Yet n-gram models fail to model this correlation for $n < 8$.

In other example, a bigram model would probably choose "blue sea" over "blue see." We would probably do the same lacking additional information about the sentence. But what if we knew that the sentence began with "The men in blue...?" We would then prefer "blue see" over "blue sea." Our preference seems to be based on the fact that we assume that "men in blue" is the subject of the sentence and we now expect a verb that this subject is likely to perform. n-gram models have no way of capturing this structural information, and the subject-verb relationship takes place over too many words to be considered by bigram models.

In spite of these drawbacks, the models with best theoretical and practical performance today all have n-gram components and it appears to be extremely difficult to create good language models without them.

2.3.2. Class based models

Class based models were introduced to alleviate some of the sparse data problems associated with trigram models. These models are based on the observation that some words behave similarly to others in certain contexts. If they could be treated together in the same manner as if they were the same word, the number of words and thus the number of different possible trigrams would decrease.

Assume we have some way of mapping a word w_i into a class c_i. An *n-gram class model* is an n-gram model on classes that assigns

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = P(w_i|c_i)P(c_i|c_{i-n+1}, \ldots, c_{i-1}).$$

Similar to the calculations for n-gram models, probabilities are determined by counts in the training data. Of course class-based systems need not be limited to n-gram class models: conditioning on combinations of words and assigning classes probabilisticly rather than deterministically is quite common.

In [19], part-of-speech assignments are used to assign classes in a trigram class model and it appears that when combined with a regular trigram model by linear interpolation, the results are an improvement over the combined model's individual components. Without adding the regular trigram model however, the class-based model does not perform as well. One reason is that the part-of-speech categories used were quite limited. Other, more general problems with part-of-speech based clustering include:

1. It might be difficult to acquire training data with part-of-speech assignments for certain domains. While there are automatic methods to derive part-of-speech assignments [19, appendix C], [11], [5], these systems are not perfect and on certain domains such as continuous conversational speech, they perform quite poorly.
2. It is unclear whether clustering by part-of-speech is suited for language modeling [19]. Language modeling might benefit more by other groupings of words.

Instead of using preassigned classes, words can be placed into classes based on other criteria such as minimizing loss of mutual information [19, appendix D] [6]. A trigram class model that grouped words in this manner did not perform quite as well as a word-based trigram model, but the decrease in the percentage of unseen n-grams in previously unseen data was substantial, from 14.7% to 3.8%. The amount of storage needed for the class-based model was also substantially (1/3) lower than needed for the word-based trigram model [6].

While class based models do alleviate some of the problems with n-gram models at the expense of slightly poorer performance, they still do not address the issue of making predictions using longer range and structural information.

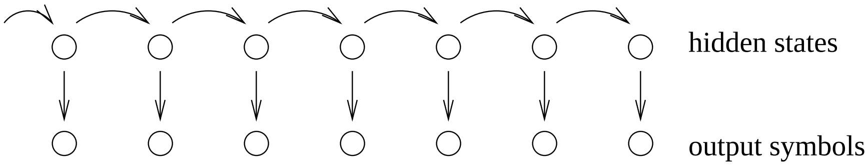


Figure 2: A HMM as a network

2.3.3. Hidden Markov models

Hidden Markov Models (HMMs) are formal systems that have applications to language modeling. We describe them here because n-gram systems, both class and word based, can be placed within the HMM framework.

A hidden Markov model consists of a set of states $S = \{s_1, s_2, \ldots, s_\sigma\}$, a probability function $\pi : S \to [0,1]$, an alphabet $W = \{w_1, w_2, \ldots, w_\omega\}$, a set of state transition probabilities $\{a_{s,t}\}_{s,t \in S}$, and a set of output probability functions $\{b_s : W \to [0,1]\}_{s \in S}$ with the requirements that

$$\begin{aligned} \sum_{s \in S} \pi(s) &= 1, \\ \sum_{t \in S} a_{s,t} &= 1, \quad \forall s \in S, \text{ and} \\ \sum_{w \in W} b_s(w) &= 1, \quad \forall s \in S. \end{aligned}$$

An HMM can be thought of as a probabilistic finite state machine that begins in state s with probability $\pi(s)$, outputs symbol w with probability $b_s(w)$, moves to state s' with probability $a_{ss'}$, outputs symbol w' with probability $b_{s'}(w')$, and so on.

Natural language can be modeled with HMMs. Assume that we have a corpus of text and we would like to find the HMM that models the language used in the corpus. We can try to find the HMM that is most likely to produce the corpus as its output. We assume that this (as yet unknown) HMM runs through its states producing output symbols that correspond to the corpus text. The HMMs states are known as *hidden* as they are not known and the output symbols (the words in the corpus text) are called *observed.* This is illustrated in Figure 2. Note that this corresponds to our definition of HMMs in that the output symbols depend only on the current (hidden) state and the state transitions depend only on the previous states.

There are efficient algorithms [41] that attempt to determine the parameters of the HMM ($a_{s,t}$, $b(s)$, and $\pi(s)$) that maximize the likelihood of a given observed sequence. Unfortunately the algorithms can get stuck in local maxima. Another, more crucial issue that needs to be considered when using HMMs for language modeling is the *Markov assumption* they make, meaning that they are memoryless; all predicative power comes from the current state. Language models such as n-grams that utilize history can be placed in the HMM framework by increasing the size of the state space but higher order interactions among states might be needed for more sophisticated models. (See section 5.3 below.)

2.3.4. Cache-based models

In section 2.1 we mentioned that language models are typically restricted to a subset of English, often in a certain domain. Training a language model specifically for that domain will cause it to perform better in that domain than a more general model would. We can take this idea of focusing our language models on specific domains and generalize a bit: we can try and build a model that will learn information about the domain and improve its predicative power as it is used. This would be useful for both domain specific and more general language models.

Caches can be used to dynamically modify the parameters of a language model to make it more specific to the text it is given. A number of cache-based techniques have been used such as creating caches of recently used trigrams [23], creating unigram caches based on part-of-speech information [25] [26], and creating selective unigram and bigram caches for words [42]. All of these cache-based systems improved on their non-cache counterparts.

The use of caches results in longer range dependencies than n-gram models by allowing words to enter the cache and affect the likelihood of future words. Caches thus allow information from the history not used in the other models seen until now to affect the choice of words. Cache models still don't utilize much syntactic information in the history to build their predictions, but they do appear to use more of the considerations that humans use in the Shannon game.

In the next section we will look at *triggers*, which can be viewed as an extended notion of a cache.

2.3.5. Trigger models

In [42], Rosenfeld demonstrated that triggers can be a useful component of a language model. He defined $(A \rightarrow B)$ to be a *trigger pair* if the word sequence A is significantly correlated with the word sequence B. For a trigger pair $(A \rightarrow B)$, A is known as the *trigger* and B the *triggered sequence*. Rosenfeld used trigger pairs to extract long-distance information from a document's history by changing the probability of B's occurrence when A appears in the document. The change in B's probability can be influenced in a number of ways: The number of times A occured in the history or it's location in the history (or both) might be factors in determining the extent of the change. Or one can choose to ignore the quantity and location of the triggers and have B's probability just be influenced by the trigger appearing at least once, somewhere in the history. Rosenfeld used this latter method and also limited A and B to be single words.

Triggers can be thought of as an extended notion of cache. In cache-based systems, a word enters the cache if it appears often enough in recent history. Triggers allow words to enter a cache even if they have not yet appeared in the text. This can be useful as we expect that certain words are likely to appear if they are highly correlated with other words that have appeared. While this generalization of cache is useful, it turns out that self-triggers (of the form $A \rightarrow A$), which could be modeled with a regular cache, are often among the most beneficial triggers.

Rosenfeld uses average mutual information between the events

$$\begin{aligned} W &= \{W \text{ is the next word in the text}\} \\ W_h &= \{W \text{ appeared in the history of the document}\} \end{aligned}$$

to find trigger pairs with significant expected benefit. On a corpus of financial news, a model combining triggers with trigrams, bigrams, and unigrams reduced perplexity by 25-27% over a regular trigram language model.

Triggers are not limited to pairs ($A \rightarrow B$) whose components actually appear in the text. For example, Rosenfeld also experimented with class-based triggers in which semantic baseforms trigger other semantic baseforms. Any occurrence of a word with the baseform A would trigger all the words with baseform B. While this model did not fare as well as the previous one, using triggers that incorporate more information than just word pairs might be beneficial. In section 7.2.1 we describe one such experiment.

3. Thesis statement

Language models use various information sources to rank choices of words in a sentence as being more or less likely given a history. The choice of information sources and how they are used plays a crucial role in determining how well a word can be predicted. Some models such as n-gram and class-based models utilize only limited context and only local information towards this goal. If more context and more information from a document's history were brought to bear, we would expect a model's predicative powers to improve. The Shannon game supports this conclusion—humans are extremely adept at predicting future words and it appears that the factor in their predictions are words' histories.

The information we use from a word's history must be chosen with care. We can not use the entire history because of the sparse data problem and imprudent choices of historical information would do little to improve our models while incurring the expense of additional parameters. If we could, we would like to capture the components that we believe humans rely on. While we do not know exactly what all of these components are, we expect that some of them are related to syntactic and semantic considerations. While we don't expect to achieve the predicative power of humans, by utilizing these considerations in language models, we hope to improve over the models in section 2.3, which contained only very limited semantic and syntactic restrictions.

Our thesis is that **incorporating syntactic structure as an information source will lead to improved language models**. Syntactic structure can place restrictions on word choice that are not accounted for by the other language models discussed. We intend to demonstrate the veracity of our thesis by building language models that utilize syntactic structure in addition to other information sources in order to more accurately predict a word given its history.

In order to achieve this goal we need to address a number of issues and questions:

1. How will we discover syntactic structure in our training data and use it in language modeling? This question addresses three related issues. The first is the basic question of our ability to syntactically parse our training data. The second relates to the fact that even if we are able to parse the data, the information produced might be in a form that is either difficult to use or not very useful in language modeling. The third addresses the issue of what syntactic structure produced by the parser should actually be used by our language models. Using all of the syntactic structure produced would result in the sparse data problem.

2. What other information sources should we utilize in our language models? Basing our mod-

els on syntactic information alone would probably not be sufficient for improvement. The sentence "Colorless green ideas sleep furiously" introduced by Chomsky demonstrates this quite admirably. The sentence is grammatical yet is highly unlikely to appear in discourse not related to linguistics. The crux of the matter is that the sentence is not semantically meaningful. Statistics gathered from parsing a training corpus can provide some grammatical restrictions on sentence semantics but many additional restrictions could be added by incorporating other information sources.

3. How will we combine the different information sources we will be using into one language model? How will we train the model's parameters?

4. On what texts will we train and ultimately test our language models? This relates to the first issue discussed—we must be able to syntactically parse the text we use.

5. How will we measure the accuracy of our models? How will we determine whether or not our models improve on existing ones?

We will discuss some of these issues in greater detail in the next few sections as we develop our research plan.

4. Grammatical formalism

In this section we address some of the issues related to discovering syntactic structure in our training data and using it in language modeling. We will first explore using probabilistic context-free grammars (PCFGs) as a method of using the syntactic information provided by context-free grammars (CFGs). Upon examination, we will see that the PCFG formalism does not lend itself to modeling lexical context in a natural way. We will propose using a robust version of link grammar as our grammatical formalism and explain how it overcomes the shortcomings of context-free grammars.

4.1. Probabilistic context-free grammars

Context-free grammars [8] [9] [10] enjoy widespread use in the study of natural language. CFGs are able to model hierarchical data and the formalism has been used to create syntactic parsers for natural language. Given a CFG for a language, one can build a stochastic grammar that can be used to model that language.

A PCFG is essentially a CFG with probabilities attached to its rewrite rules. Formally, a PCFG $\mathcal{G}$ is a pair (G, P) where $G = (V_N, V_T, S, R)$ is a CFG with nonterminal symbols $V_N = \{V_N^i\}_{i=1}^{n_{V_N}}$, terminal symbols $V_T = \{V_T^i\}_{i=1}^{n_{V_T}}$, initial state $S \in V$, and rewrite rules $R = \{R^i : V_N \rightarrow (V_N \cup V_T)^+\}_{i=1}^{n_R}$. $P : R \rightarrow [0,1]$ is a probability function that satisfies

$$\begin{array}{lll} 1. & P(A \rightarrow \alpha) \geq 0, & \forall (A \rightarrow \alpha) \in R \\ 2. & \sum_{1 \leq j \leq n_i} P(R_{i_j}) = 1, & \forall i, 1 \leq i \leq n_{V_N} \end{array}$$

where R_{i_j} is the j-th rule in R with V_i as its left-hand-side and n_i is the number of such rules. The probability of a rule, $P(A \rightarrow \alpha)$, is the probability of generating α given the nonterminal A. We will therefore sometimes write $P(\alpha|A)$ for $P(A \to \alpha)$. This notation will prove convenient when we discuss the probability of a derivation.

Given a sequence of words $W = w_1, w_2, \ldots, w_N$ from V_T, we can compute the probability that it is generated by $\mathcal{G}$. As many context-free grammars are ambiguous, more than one parse tree can correspond to W. We sum over the different trees to account for the different ways to parse the word sequence. We will denote by $\mathcal{T}$ the set of parse trees corresponding to W. The probability that W is generated by the grammar $\mathcal{G}$ will be

$$P(W) = \sum_{T \in \mathcal{T}} P(W, T)$$

where $P(W, T)$ is the joint probability of generating W and T.[1] The probability $P(W, T)$ is simply the product of the probabilities of all the rules $A \to \alpha$ such that A is the parent of α in T.

4.1.1. PCFGs for language modeling

The PCFG formalism has many features that make it attractive for language modeling:

1. There are many natural language grammars already written for the CFG formalism. We would be able to utilize one and not have to write one from scratch.
2. There are *treebanks* of parsed data which we could utilize to train and test our model.
3. The CFG formalism is well understood and for simple grammars it is often easy to see the correspondence between a parse tree and the syntax of a sentence.
4. Robust parsers based on the CFG formalism exist [32] [37] [30] [31]. Robust parsing is necessary as language models must assign a probability to any input string. Except in extremely restricted domains, *any* input is possible. While some inputs might be highly unlikely, they are still possible and language models must still consider them as possibilities.
5. Algorithms exist to compute $P(W)$, the probability that the grammar generates word string W [21]. This allows us to rank hypothesized sentences as being more or less likely.
6. Algorithms exist to find the k most likely parse trees that correspond to a sentence [21].
7. Algorithms exist to modify rule probabilities in order to maximize the probability of generating a corpus [2] [21]. This allows us to use a corpus to train the model's parameters.
8. Algorithms exist to compute prefix probabilities, the probabilities of initial word strings [20]. This might have some advantages when the language model is used in systems such as speech recognition systems where left-to-right processing of the input is useful.

Some might argue that some of these features are limited in usefulness. For example, some of the robust parsers have limitations in speed [32] [37] [31] or language [30], and computing the

[1] We are slightly abusing notation here. The probabilities should really be conditioned on the grammar $\mathcal{G}$ and in addition, we are using P to denote a number of different probability functions. As we are not changing grammars and the probability function should be clear from the context, we choose to simplify notation.

probability of an initial word string requires the inversion of what can be a very large matrix [20].[2] Also, some of the features above might not have analogues for robust PCFG-based parsers. Still, these features certainly give us good reason to consider using PCFGs to provide us with the syntactic structure we expect will lead to improved language models.

Unfortunately, as we defined PCFGs above, their usefulness in language modeling is extremely limited. In order to justify this statement, it is useful to view the probability of a word string within the context of the string's derivation. We would like the probability of a derivation to be equal to the probability of the parse tree it produces however many derivations can correspond to the same parse tree. We will therefore concentrate on derivations known as *leftmost derivations* in which rules are only applied to the leftmost non-terminal in a derivation string. As there is a 1-1 correspondence between leftmost derivations and parse trees, we can now consider the probability of a leftmost derivation and its corresponding parse tree to be the same. Assume a leftmost derivation of W

$$S \overset{A_1 \to \alpha_1}{\longrightarrow} \beta_1 \overset{A_2 \to \alpha_2}{\longrightarrow} \beta_2 \overset{A_3 \to \alpha_3}{\longrightarrow} \cdots \overset{A_m \to \alpha_m}{\longrightarrow} W \tag{2}$$

in which rule $A_i \to \alpha_i$ is applied at time i. The joint probability of W and the corresponding parse tree is

$$\prod_{i=1}^{m} P(A_i \to \alpha_i) = \prod_{i=1}^{m} P(\alpha_i | A_i). \tag{3}$$

We see that when computing the probability of a parse tree and hence a word string, rules are invoked with probabilities that are conditioned only on the left-most nonterminal in the current derivation string. That is, as described, PCFGs make a Markov assumption—probabilities depend only on the nonterminals being rewritten and not on any surrounding context. This assumption severely limits the lexical context used by language models within this framework.

From what we know about language modeling, lexical context appears to be crucial for good language models. Lexical context alone is responsible for the success of trigrams models. When a language model is actually used, other than what is encoded in the parameters of the model itself, the only information available to help in predicting a word are the words that preceded it. But in PCFGs, the previous words have no influence on the choice of rules at all! While we believe that syntactic structure will improve language models, the structure must not come at the expense of ignoring restrictions imposed by the lexical components in the sentence.

In [36], a language model that combines both bigrams and a stochastic context-free grammar is presented. The relative frequencies of the bigrams are imposed as constraints on the stochastic context-free trees. This approach is important as it attempts to integrate lexical considerations with grammatical structure. The bigram constraints attempt to alleviate the deficiencies of the CFG. The method is currently not practical on a large scale as random sampling is needed to draw sample word strings according to the constrained distribution.

4.1.2. PCFGs—second attempt

In turns out that we can modify the definition of PCFGs in order to capture additional lexical context. When we wrote equation (3) for the probability of the derivation (2), each term in the product was conditioned on only one element of the derivation string existent at that point.

[2] The size of the matrix is $n_{V_N} \times n_{V_N}$ where n_{V_N} is the number of non-terminal symbols in the grammar.

We limited the dependence in order to remain true to our previous definition of PCFGs. By conditioning on more of the derivation string and appropriately modifying the definition of PCFGs we can achieve not only a greater degree of lexical context, but we can allow other aspects of the derivation string such as syntactic relations to influence the derivation. Essentially we can achieve context-free grammars with context-sensitive probabilities.

For the PCFG $\mathcal{G}$ we will still define $P(W) = \sum_{T \in \mathcal{T}} P(W,T)$ as before, but we will now define $P(W,T)$ for a tree T with leftmost derivation (2) as $\prod_{i=1}^{m} P(\alpha_i | c[\beta_{i-1}])$ where $\beta_0 = S$ and $c[\cdot]$ assigns histories to equivalence classes.

A number of CFG-based parsers that incorporate additional context have been built. For example, in [34], the probability of a rule was conditioned on the immediate parent of the rule and on limited part-of-speech information. The grammar used in this work differs from a PCFG in the method it uses to score parses. While the trained parser was able to achieve fairly good accuracy on test data, the results might not be indicative of how well such a model would work in general as the experiments were run on a relatively easy domain, only limited training and test data were used, and the test data might have influenced the parser's development. In [35] a similar model was examined and the results reported were not as good.

Black *et. al.* [4] described *history-based grammars* (HGBs) in which rich histories are used to aid in prediction. Essentially, any information previously seen is allowed to influence the expansion of a non-terminal node. Decision trees were used in this work to estimate the rule probabilities. The authors tested a HBG and found it to be more accurate than a PCFG it was compared to. Another decision tree based model was used by Magerman in [33] with good results.

While some of these systems seem quite adept at finding good parses of input sentences, they can not be directly used as language models as they don't calculate the prior probabilities we are interested in.

Unfortunately, we know of no examples where language models based on PCFGs, even with context-sensitive probabilities, performed better than n-gram models except for on small, restricted domains. While we can't be *completely* sure why no one has succeeded in improving language models by including a PCFG component (and we are fairly confident that many have tried), we will mention some of the reasons why we choose not to make the attempt.

1. It is difficult to decide what equivalence classes to use. While this is a problem of language modeling in general, it seems to be more acute here. We would certainly hope to condition on local lexical information, but how do we determine what other information to use?

2. As it is difficult to decide which elements from the history to condition on, one might be inclined to condition on a large history. Even relatively small histories, however, can lead to the sparse data problem. For example, in [35], the probability of a rule $A \rightarrow \alpha$ is conditioned on A's parent and the part-of-speech trigram centered on the leftmost word that A dominates. Parts-of-speech rather than words were used for prediction because of the sparse data problem.

3. Some CFG parsing algorithms such as the CKY algorithm are quadratically dependent on the size of the grammar (and cubic in the number of tokens being parsed). The quadratic dependency comes from being able to treat non-terminals with the same label that span the same part of the sentence as being identical. When conditioned on elements from the history, non-terminals can not always be treated the same and this dependence no longer holds. While

work has been done to overcome this limitation [35], new restrictions need to be placed on the model to make parsing practical.

4. All of the models that we are aware of that attempt to incorporate non-trivial historic information were trained from treebank data. It is unclear if it would be feasible to train these systems from unmarked corpora, which would be necessary in order to achieve good performance on many domains for which no treebank data exists.

5. What might be the most important reason of all is that the context-free grammar formalism just doesn't appear to lend itself to modeling lexical context in a natural way. The "standard" model of probabilistic context-free grammar does not work and one must make significant modifications and compromises in order to achieve the high degree of lexicality that simplistic language models such as trigrams achieve.

We must find language models that are better suited than PCFGs for incorporating lexical context. In addition they must contain syntactic structure we can utilize to place restrictions on word choices. The models must be based on a formalism that lends itself to extracting relevant information from the data its grammars generate. We propose basing our language models on *link grammars* [44].

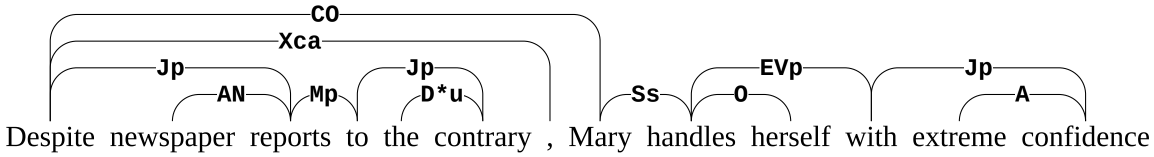
4.2. Link grammar

A formal grammatical system called link grammar was introduced by Sleator and Temperley in [44]. Although the expressive power of link grammars is equivalent to that of context-free grammars, Sleator and Temperley believe that the encoding of natural language grammars is much easier in their system. One reason for this is that the grammar is described at the lexical level; one describes how words behave without forcing them into preconceived categories based on e.g. part-of-speech information. Link grammar allows both a high degree of lexicality and syntactic structure which makes it a good candidate for our language models. In section 4.2.1 we will review link grammar, in section 4.2.2 we will briefly argue some of its merits for language modeling, in section 4.2.3 we will discuss a robust parser based on the link grammar formalism, in section 4.2.4 we will review some work on probabilistic models of link grammar, and in section 4.2.5 we will briefly discuss some of the approaches we plan on taking in building a link grammar-based language model.

4.2.1. Link grammar formalism

In this section we briefly summarize the relevant concepts and notation of link grammar.

The figure below represents one of the parses produced by a link grammar parser on the input sentence "Despite newspaper reports to the contrary, Mary handles herself with extreme confidence." The labelled arcs connecting words to other words on their left or right are called *links*. A valid parse is called a *linkage*.



In this figure we see that `confidence` is linked on the left to the adjective `extreme` with a link labelled `A`, denoting an adjective. (We can also say that `extreme` is linked on the right to `confidence`.) Words can have multiple links; for example, The word `handles` has a singular noun (`Ss`) link to its left and object (`O`) and prepositional phrase (`EVp`) links to its right.

A link grammar is defined by a dictionary comprising a vocabulary and definitions of the vocabulary words. These definitions describe how the words can be used in linkages. A word's definition can be thought of as a list of *disjuncts*. Each disjunct d is represented by two ordered lists, written as

$$d = (\,(l_1, l_2, \ldots, l_m)\ (r_n, r_{n-1}, \ldots, r_1)\,)$$

where l_i are *left connectors* and r_j are *right connectors*. (Connector names typically begin with one or more upper case letters followed by sequences of lower case letters and *'s.) For example, from the above linkage we can see that one of the disjuncts in the dictionary definition of `handles` must be ((`Ss`) (`EVp`, `O`)).

A word W with disjunct $d = (\,(l_1, l_2, \ldots, l_m)\ (r_n, r_{n-1}, \ldots, r_1)\,)$ can be linked to other words by connecting each l_i to right connectors of words on W's left and also connecting each r_j to left connectors of words on W's right. Links are only permitted between *matching* connectors.

A linkage is specified by choosing a disjunct for each word in the sentence from the word's definition and linking every connector in the disjunct with a connector of a different word's disjunct. A linkage is valid if and only if it meets the following criteria:

- Connectivity: The graph of links and words is connected.
- Planarity: When drawn above the sentence, the links do not cross.
- Ordering: For disjunct $d = (\,(l_1, l_2, \ldots, l_m)\ (r_n, r_{n-1}, \ldots, r_1)\,)$ of word W, l_i and r_j are linked to words whose distances from W are monotonically increasing in i and j.
- Exclusion: No two links connect the same pair of words.

While the use of disjuncts simplifies the mathematical analysis and parsing algorithms for link grammars, it is cumbersome to express actual grammars in these terms. For this reason the linking requirements of a word are often expressed as a logical formula involving connectors and the operators `&` and `or`. As a simple example, the formula

```
(D- & (O- or S+))
```

represents the two disjuncts ((`D`) (`S`)) and ((`D`, `O`) ()). The links attached to each word in a linkage must *satisfy* that word's formula. That is, the connectors must make the logical expression represented by the formula true, with the `or` operator understood to be exclusive.

4.2.2. Link grammar for language modeling

In [44] it was shown that link grammars have the same expressive power as context-free grammars. Yet in spite of our rejection of PCFGs as our basis for language modeling, we choose to go forward with link grammars. The key reason is that link grammars are highly lexical whereas context-free

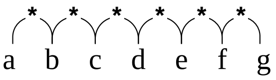


Figure 3: A bigram linkage

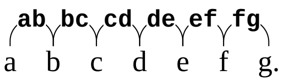
grammars are not. Context-free grammars primarily deal with symbols, words only come into play at the leaves of the parse trees. Link grammars on the other hand, use the words of the sentence throughout the parsing process, linking words of the sentence with one another.

We previously saw that n-grams are used in practically all language models today and that we believe it difficult to create a good language model without an n-gram component. Fortunately, the lexicality of link grammar allows n-grams to be easily incorporated into its framework [28]. For example, Figure 3 represents a *bigram linkage* of a sentence. The simple link grammar corresponding to linkages of this form can be trained [28] into a probabilistic model that corresponds with the regular bigram language model.

Trigram language models can also be easily attained by training a link grammar that contains the following disjuncts for each word w,

$$\begin{gathered} (\,(\textsc{Nil})\ (wx)\,) \\ (\,(xw)\ (wy)\,) \\ (\,(xw)\ (\textsc{Nil})\,), \end{gathered}$$

where x and y run over all the words of the vocabulary. Linkages in this grammar will look like



While bigram and trigram models can also be built using the mechanics of PCFGs, it is unclear how to cleanly incorporate the productions used for these models into existing context-free grammars or how to incrementally add grammatical constructs onto existing PCFG n-gram models. Owing at least in part to the lexicality of link grammars, combining n-gram disjuncts with "grammatical disjuncts" is relatively easy. In fact, grammar can be slowly added to an n-gram link grammar, allowing the grammatical disjuncts to account for some of the links in the sentence leaving the rest of the sentence to be handled by the n-gram disjuncts. This allows for experimentation to see how various grammatical constructions affect the performance of link-grammar based language models.

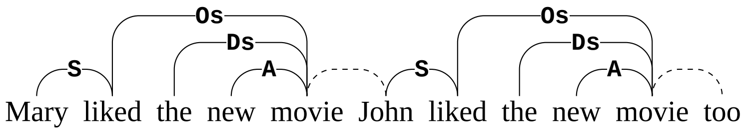
Rather than beginning with a link grammar that is equivalent to an n-gram model and slowly adding grammar, one might start with an existing link grammar and allow portions of the sentence that it can't parse to be handled by n-gram components. With this as one of our objectives, we designed a robust link grammar parser that parses using a combination of regular disjuncts and local links.

4.2.3. A robust link grammar parser

The approach we took to robust parsing uses the notion of *cost*. Informally, the idea is to define a set of generalized parse trees with respect to the grammar and the assignment of a number $cost(P|S) \geq 0$ to each generalized parse P of a sentence S. The task of the robust parsing algorithm is to find all generalized parses having minimum cost.

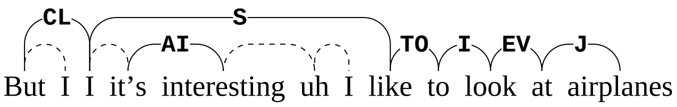
Our robust parser uses the concept of a *null link* to allow connections between any pair of adjacent words, regardless of their dictionary definitions.

The following linkage has two null links, drawn as dashed arcs.



If both of these null links are deleted, then we obtain two legal linkages together with the disconnected word "too." If a linkage satisfying the criteria of section 4.2.1 has the property that removing its null links results in a collection of one or more legal linkages together with zero or more isolated words, we say that the linkage is *chained* and is considered a generalized parse. We define a cost to each chained linkage equal to the number of null links it contains. The cost of a string of words is defined to be the minimum cost of all chained linkages connecting those words. By this measure, the cost of a string of N words can be no greater than $N-1$, and it is zero only if the string is grammatical.

In [17] we describe an efficient parsing algorithm that finds the minimal cost generalized parses of a sentence. This algorithm is able to extract relevant grammatical structure even in domains where very few of the sentences are "grammatical" by standard criteria. As an example, we show one sentence that contains an embedded phrase that is grammatical in isolation but not with respect to the larger utterance. The robust parser handles this by connecting the embedded phrase to the larger utterance with a null link.



We expect that much of our work in language modeling will be based on the robust link grammar parser. The parser allows us to use relatively small grammars and have the rest of the sentence connected by null links which could be modeled as n-gram links in a language model. Another possibility would be to use the robust parser in conjunction with the probabilistic model of link grammars developed in [28] and reviewed in the next section.

4.2.4. Previous work on probabilistic link grammars

In [28] it is observed that in the link grammar parsing algorithm a link depends on a left connector l of a word L and a right connector r of a word R. A link is formed by first choosing a word w to link to, followed by a disjunct d for that word. An *orientation* O is chosen for d as it might link to

l, r, or both. This suggests a probabilistic model of link grammar with the parameters

$$P(w, d, O|L, R, l, r).$$

This probability can be partitioned by

$$P(w, d, O|L, R, l, r) = P(w|L, R, l, r)P(d|w, L, R, l, r)P(O|d, w, L, R, l, r)$$

and approximated as

$$P(w, d, O|L, R, l, r) \approx P(w|L, R, l, r)P(d|w, l, r)P(O|d, l, r) \tag{4}$$

in order to reduce the number of parameters in the resulting model.

The probability of a linkage is the product of all of its link probabilities. For a linkage $\mathcal{L} = \{(w, d, O, L, R, l, r)\}$, the probability of generating a sentence S with linkage $\mathcal{L}$ will be

$$P(S, \mathcal{L}) = P(w_0, d_0) \prod_{\mathcal{L}} P(w, d, O|L, R, l, r),$$

where w_0 is an initial word and d_0 is an initial disjunct. A training algorithm is given in [28] that determines the parameters of the probabilistic model given a training corpus by maximum likelihood estimation using the EM algorithm.

4.2.5. Initial approaches

Given the grammar introduced in [44] and the robust parser of [17] we could use the probabilistic model and training algorithm of [28] to create a link grammar based language model. Instead, we choose to begin with simple models such as bigrams or trigrams and to slowly add grammatical structure and additional information sources to these models. There are a number of reasons to begin with this approach, including the following:

1. Basing a language model on a relatively large grammar and parameterizing it as described in [28] might lead to a sparse data problem. Grammars can contain large numbers of disjuncts, many of them rare. Obtaining reliable estimates of likelihoods might not be possible. Even if we reduced the size of our grammar by eliminating some of the less common disjuncts, it might still be difficult to estimate some of the terms of equation (4). $P(w|L, R, l, r)$, for example, depends not only on the three words L, R, and w, it also depends on the connectors l and r. The percentage of new tuples of the form (w, L, R, l, r) appearing in the test data would probably be much higher than the percentages reported in [19] [6] for trigrams. While a deleted interpolation algorithm for smoothing is given in [39], we still prefer to begin with simpler models.

2. It is unclear how to incorporate some elements of link grammars such as multi-connectors and conjunctions into the framework of [28].

3. Given the success of n-gram based language models we would like n-grams to be incorporated into our language models. We could add n-gram disjuncts to the grammar given in [44], but that would introduce numerous linkages of the sentences in the training corpus, considerably increasing training time.

4. If we begin our work by building a language model that incorporates all of the information available by parsing a corpus with a detailed grammar, it will be difficult to determine which elements of the grammar actually contribute to improved language models. For example, it would be interesting to know how noun-verb agreement affects a language model. By starting with a simple language model, we will have a better chance at getting a handle on which elements contribute to better models.

5. We might want to incorporate information sources into our language model that are not directly part of the link grammar formalism. We plan on exploring the addition of a number of elements including the following:

   - Topic dependence: It is widely recognized that certain words are more likely to appear in certain topics of discourse than others. For example, when discussing recycling, words like "cans," "paper," "plastic," "trash," and so on are more likely to appear than they would in most other contexts. If we knew that the topic of discourse was recycling, we would like our language model to increase the probabilities of these words appearing at the expense of less likely ones. We can utilize information that link grammar provides to more accurately model this dependence. For example, while "trash" is likely to appear in the context of recycling, it is far more likely to appear as a noun than a verb. Link-grammar can be used to detect parts-of-speech which will enable us to increase the likelihood of "trash/n" while not increasing the likelihood of "trash/v."
   - Caches: Even if we can't determine the topic of discourse we can use caches to capture some topic dependency and take advantage of the dynamic nature of discourse. Using information from link grammar, we can limit caches to specific parts-of-speech and/or language constructions. Grammatical information will allow us to weed out certain words we should not cache (such as function words) and help us to focus on words that are more likely to benefit from caching (such as nouns and objects).
   - Triggers: In [42], triggers were found to be a useful information bearing element to incorporate into language models and it would probably be beneficial to include them in ours as well. By utilizing link information we can examine the possible benefits of utilizing part-of-speech or other grammatical considerations when choosing trigger pairs. For example, "trash" would have quite a different set of triggered words when used as a noun than as a verb.

Not only do we not have to utilize the probabilistic model of [28] in its entirety, our model does not even have to produce valid linkages of its input text. We can utilize the data produced from parsing a training corpus as an information source and use as much or as little of it as we please. For example, one simple language model might estimate

$$P(w_i|w_1,\ldots,w_{i-1}) \quad = \quad \sum_c P(w_i|w_{i-1},c_i)P(c_i|w_{i-1},c_{i-1}) \tag{5}$$

where c_j is a connector associated with word w_j. $P(w_i|w_{i-1},c)$ might be calculated as a combination of the unigram probability $P(w_i)$, the bigram probability $P(w_i|w_{i-1})$, and the probability $P(w_i|c)$, which is similar to the probability of a part-of-speech assignment. These probabilities can be estimated by using data collected by parsing a corpus with a link grammar and can be combined to estimate $P(w_i|w_{i-1},c)$ by any number of methods such as *deleted interpolation* [22]. We can calculate $P(c|w_{i-1},c_{i-1})$ in a similar fashion. This model, which is reminiscent of the part-of-speech class models we reviewed in section 2.3, uses very little of the information available from the parsed training data and when viewed as a generative model, does not produce linkages at all! The point here is that just because we are using link grammar to parse our training data, our language model is not required to adhere to the link grammar formalism itself.

5. Combining Information

Many of the language models we will use will be comprised of a number of components, somehow combined together to give the probability of a word given its history. For example, we mentioned that for the simple model of (5), $P(w_i|w_{i-1}, c)$ might be estimated by some combination of $P(w_i)$, $P(w_i|w_{i-1})$, and $P(w_i|c)$. How should these components be combined to form one integrated estimate?

5.1. Deleted interpolation

Deleted interpolation [22] has traditionally been used to alleviate sparse data problems. More detailed models (such as trigrams) are usually combined with less detailed ones (such as bigrams and unigrams) by linear interpolation. The coefficients of the combined model are trained using the forward-backward algorithm on unseen data. Not limited to solving sparse data problems, deleted interpolation can also be used to combine any number of language models into one integrated scheme.

While deleted interpolation can be used to combine our language model components, we will be using a different method, which uses a maximum entropy approach [42] [3] [40].

5.2. Maximum entropy principle

The maximum entropy method [18] chooses a model that agrees with certain explicit constraints yet avoids assuming anything that has not been specified. For example, we might know that in our training data, the noun "toad" is modified by adjective "slimy" 10 times. We might want our language model to agree with this observation on the training data, at least in expectation. In general, many different probability distributions will satisfy this criterion. Subject to the specified constraints, the maximum entropy method chooses that distribution which is "most uniform," where uniformity is defined in terms of Shannon entropy. It is certainly intuitive that we would like a model that doesn't make assumptions about things it hasn't been told.

How do we go about specifying constraints that the language model must observe? We define functions called *feature functions* on the event space (h, w) and require that the expectations of these functions are equal to the empirical expectations from the training data. This results in a set of linear relations called *constraints.*

The maximum entropy distribution subject to the constraints

$$\{E_P[f_i] = E_{\tilde{P}}[f_i]\}_{i=1}^{m} \tag{6}$$

has a particularly simple form. If $\lambda = (\lambda_i, \lambda_2, \ldots, \lambda_m)$ is a vector of real numbers, one for each constraint, it is easy to show that the maximum entropy solution to the system of equations (6) is a member of the exponential family

$$\mathcal{E} = \left\{ \ P_\lambda \mid P_\lambda(w|h) = \frac{1}{Z_\lambda(h)} e^{\sum_i \lambda_i f_i(w,h)} \ \right\}$$

where

$$Z_\lambda(h) = \sum_w e^{\sum_i \lambda_i f_i(w,h)}$$

is used for normalization [18].

The maximum entropy solution has a property that is intuitively appealing. Often in language modeling one seeks the probability distribution that maximizes the likelihood of the training data. While the maximum entropy solution to the system of linear constraints does not quite satisfy this property, it satisfies a similar one. Among all of the elements of the exponential family $\mathcal{E}$, the maximum entropy distribution is also the maximum likelihood solution.

The maximum entropy principle has been successfully used as a method to combine information sources in natural language applications [42] [3] [40] and has a number of appealing properties:

1. The principle is simple, intuitive and easily allows one to combine multiple information sources. The maximum likelihood dual seems to lend additional evidence that the maximum likelihood solution is a natural one to use.

2. A number of numerical algorithms (including generalized iterative scaling [13] and improved versions of it [3] [40]) exist for computing the parameters λ_i of the maximum entropy solution The maximum entropy solution is unique, and these algorithms are guaranteed to converge to it.

3. If additional information sources are added after training a model, an adequately performing model might be achieved by training the parameters for the new information sources while keeping the others constant. While this won't result in the maximum entropy solution, the result might still be good enough for many purposes. This method was used to incrementally add constraints to models in [3] and [40].

One item that is lacking from maximum entropy models is the notion of hidden data, previously seen in section 2.3.3. In the next section we will discuss a framework for constructing probabilistic automata using maximum entropy distributions.

5.3. Gibbs-Markov models

Gibbs-Markov models (GMMs) [27] are a framework that can be used to build what can be viewed as extended hidden Markov models that allow context-dependent probabilities, thus overcoming one of the severe limitations of HMMs. GMMs are also closely related to maximum entropy models and they combine advantages of both the hidden Markov and maximum entropy frameworks. From the HMM world they have the ability to condition on hidden states such as part-of-speech information. In fact, GMMs contain *embedded* HMMs that act exactly like regular HMMs when the context-dependent part of the model is kept fixed. Closely related to maximum entropy models, restrictions on the transition and output probabilities are specified using feature functions, easily allowing context to influence state and output symbol predictions while not assuming unspecified

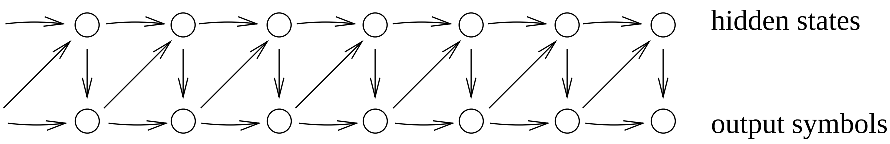


Figure 4: A simple Gibbs-Markov network

restrictions. The state and output probabilities are modeled as Gibbs distributions, members of the family in equation (5.2).

Figure 4 shows a representation of a possible GMM. This GMM corresponds to the simple language model of (5). The *hidden model*, the probability distribution that generates states, is conditioned on the previous state (as in HMMs) and the previous output symbol. The *output model* is conditioned on the current state and the previous output symbol. Each of these models is described by feature functions. For example, if (w_1, w_2) is a bigram in our training data, one of the features of the word model would be

$$f_{w_1,w_2}(w) \quad = \quad \begin{cases} 1 & \text{if } w = w_2 \text{ and the previous word was } w_1 \\ 0 & \text{otherwise} \end{cases} \quad .$$

In section 7.1.1 we include a complete description of the features of this simple language model.

GMMs are trained by using an expectation-maximization (EM) algorithm [14] where the M-step uses generalized iterative scaling for conditional maximum entropy models.

6. Testing the model

How can we judge the performance of a language model? How do we determine whether one language model is superior to another? In order to test language models, a corpus is usually divided into two parts, *training data* and *test data*. We already discussed the training data—it is used to adjust the parameters of the language model in order to provide good approximations to the terms in equation (1) for the sub-language being used. The test data is used to determine how well the language model actually performs on a specific input.

In section 6.1 we will describe the corpus we will be using in our experiments and in section 6.2 we will discuss measures of goodness for language models.

6.1. Corpus

We will be using the *Switchboard* [15] corpus of conversational English divided into training and testing data. This corpus is comprised of approximately three million words of text, corresponding to more than 150 hours of transcribed speech collected from telephone conversations restricted to 70 different topics. The Switchboard corpus was created to allow standard evaluations and provide a stable research environment for large vocabulary continuous speech recognition. The corpus contains an abundance of phenomena associated with conversational language: non-standard words, false starts, stutters, interruptions by other speakers, partial words, grammatical errors, changes in grammatical structure mid-sentence, hesitation sounds, and non-speech sounds. Relatively few of the sentences in this corpus are grammatical by any standard criteria; indeed, only 28% of the sentences could be parsed with zero null links using our robust link grammar parser.

6.2. Comparison of language models

How is the test data used to judge the worth of a language model? Two methods are commonly used to measure the goodness of a language model.

The first measure, *perplexity*, is a statistical measure that assigns a numeric value to a language model when evaluated on a test corpus. When the vocabularies and test data are the same, perplexity can be used to compare different language models to each other. The second measure, *error rate*, is extremely intuitive. Language models don't work alone in vacuums, they work in conjunction with other parts of larger systems, such as those performing speech recognition. Error rate measures how many errors the entire system makes at the task it was designed to perform. We will now explain these two measures in greater detail.

6.2.1. Perplexity

Suppose a stochastic process outputs words (symbols) $w_1, w_2, \ldots$ taken from a finite vocabulary (alphabet) according to some probability distribution P. The *entropy* of the information source is defined as

$$H(P) = \lim_{N\to\infty} -E_P \log P(w_1, w_2, \ldots, w_N)$$

where E_P denotes expectation with respect to P. We will take the logarithms to the base 2 and we will measure entropy in units called *bits*. It is a well known theorem of information theory [12] that an average of H bits is needed to represent a symbol from a source with entropy H. Entropy is a measure of the uncertainty of the source; it measures how much information, on average, is needed to describe a symbol. Entropy can therefore be viewed as the amount of information contained in the source; the higher the entropy, the greater the information.

The source with the greatest information is the one that can output L words, all equally likely, and it has entropy $\log L$. Therefore, a source with entropy $H(P)$ contains the same amount of information as a source with $2^{H(P)}$ symbols, all equally likely.

If the source is stationary and ergodic, then on any sample of words $w_1, w_2, \ldots$ drawn according to the distribution P,

$$H(P) = \lim_{N\to\infty} -\frac{1}{N} \log P(w_1 w_2 \cdots w_N). \tag{7}$$

If P is unknown but we have a sample of words $w_1, w_2, \ldots$ drawn according to P, we can compute an upper bound on $H(P)$ by modeling P with a stationary ergodic process M and computing the *cross-entropy* of P measured by M which is equal to

$$H(P, M) = \lim_{N\to\infty} -\frac{1}{N} \log M(w_1 w_2 \cdots w_N). \tag{8}$$

$H(P, M)$ is the average uncertainty of the source based on the model M. $H(P)$ measures the average uncertainty of the source itself. The difference between $H(P)$ and $H(P, M)$ measures how accurately M models P. Although $H(P)$ is not known, as $H(P)$ is bounded from above by the cross entropy, models producing lower cross entropy model P better.

How does this relate to language modeling? While it is probably not the case, we can view a stationary ergodic source as having generated the test data. This source corresponds to the distribution P above. We can then measure an approximation to the cross entropy of P as measured by our model using equation (8), ignoring values of N greater than the number of words in the test data.

Closely related to the cross-entropy of P as measured by a language model M is the *perplexity* of the model relative to P defined by

$$PP(M) = 2^{H(P,M)}.$$

The cross-entropy $H(P, M)$ is the average uncertainty in each word based on the language model M. This uncertainly is equivalent to that of a source with $PP(M)$ symbols that generates each one with equal probability.

Perplexity gives us the ability to easily compute measures of worth associated with language models. Models with lower perplexity should be considered "better" models. While a system employing a language model with low perplexity is not guaranteed to perform better than one whose language model component has higher perplexity, it is often the case that it will, particularly when the difference in perplexity is large. In any case, perplexity should certainly be used as an evaluation criteria when developing language models.

6.2.2. Error rate

Error rate is a measure of how accurately a system performs. Unlike perplexity, This measure tests a language model in the way it will actually be used. For example, in a speech recognition system, the number of errors made by the system on a standard set of spoken sentences could be used to compare system accuracy. While this in some sense is the ultimate test of a language model, it still has some problems.

One problem is that language models are only parts of larger, complex systems, in which many constituents are developed concurrently. The interactions among those components are often complex and difficult to determine. While one could keep all parts of the system except the language model static and try to improve the language model based on error rate, when changes are eventually made to other portions of the system, they might affect the performance of the language model. Another problem with using error rate is that it is a time-consuming process; it can take much longer to calculate error rate than perplexity. Using error-rate as a basis for language model development would be a slow process. Finally, one might want to work on improving language models independently of how they might be used in the future.

7. Thesis work

We now begin to describe some approaches to designing link grammar based language models. Some relatively simple models have already been tested and a few of the future steps are already

n	*bigram perplexity*	*trigram perplexity*
3000	75.0	64.1
4000	79.4	68.1
5000	82.6	70.9
8000	89.6	77.1

Table 1: Bigram and Trigram perplexities of the Switchboard corpus using the n most frequent words as the vocabulary

laid out. Other work is still in the planning stages. In this section we describe what we have accomplished and what remains to be done.

7.1. Current results

In this section we discuss some of the work and experimental results we have to date.

In order to use link grammar as a source of information about the Switchboard corpus, we need to be able to parse the corpus. With this objective, we built a robust link grammar parser, discussed in section 4.2.3 and described in greater detail in [17]. The parser is quite adept at extracting relevant structure for a large portion of the corpus even though only a small fraction of the sentences are "grammatical" by standard criteria.

One important observation to note before designing language models for the Switchboard corpus is that while trigram models demonstrate lower perplexity than bigram models on the corpus, the differences in the perplexities are not nearly as great as for a domain such as Wall Street Journal articles. Table 1 contains perplexity results for different vocabulary sizes. The closeness of the bigram and trigram perplexities leads us to build language models that, when they contain n-gram components, limit n to be smaller than three. In other words, our language models will not contain trigram components. This allows for much faster development as techniques to manage the large numbers of trigrams and their associated parameters are not needed. It also allows for faster training as many fewer parameters are involved.

7.1.1. Simple model

Our first language model is based on local information and is a Gibbs-Markov model influenced by part-of-speech and local, lexical information. This model is composed of two sub-models, a *word model* and a *connector model*. These models assume that we can associate each word of the training data with an additional datum that we refer to as a *connector*. We will, in fact, view the training data as a sequence of interspersed words and connectors of the form $c_1 w_1 c_2 w_2 \cdots$. The connector takes its name from the connectors used in link grammar to which it is closely related. The connectors correspond to the hidden layer of the GMM and the words correspond to the observed output symbols.

The word model provides estimates of the form $P(w_i|w_{i-1}, c_i)$ and the connector model provides estimates of the form $P(c_i|w_{i-1}, c_{i-1})$. In the combined model, $P(w_i|w_1, \ldots, w_{i-1})$ is calculated by using forward probabilities $\alpha_i(c)$ representing the probability of generating $w_1, \ldots, w_i$ where w_i

model	*word vocab size*	*connector vocab size*	*perplexity*
bigram model	15000	NA	99.443
model of section 7.1.1	15000	100	100.729
model of section 7.1.1	15000	1	102.732

Table 2: Perplexities for Bigram and simple maximum entropy models

has connector c. These forward probabilities are easily calculated by

$$\alpha(i+1,c) = \sum_{c'} \alpha_i(c')P(c|c',w_i)P(w_{i+1}|w_i,c).$$

The word model is a maximum entropy model with the following features:

$$\begin{aligned}
f_{w_1}(w,h) &= \begin{cases} 1 & \text{if } w = w_1 \\ 0 & \text{otherwise} \end{cases} \quad \forall\ w_1 \in V_W \\
f_{w_1,w_2}(w,h) &= \begin{cases} 1 & \text{if } w = w_2 \text{ and } h \text{ ends in } w_1, c \text{ for some connector } c \\ 0 & \text{otherwise} \end{cases} \quad \forall\ w_1, w_2 \in V_W \\
f_{w_1,c}(w,h) &= \begin{cases} 1 & \text{if } w = w_1 \text{ and } h \text{ ends in } c \\ 0 & \text{otherwise} \end{cases} \quad \forall\ w_1 \in V_W \text{ and } c \in V_C
\end{aligned}$$

The connector model is a maximum entropy model with the following features:

$$\begin{aligned}
g_{c_1}(c,h) &= \begin{cases} 1 & \text{if } c = c_1 \\ 0 & \text{otherwise} \end{cases} \quad \forall\ c_1 \in V_C \\
g_{c_1,c_2}(c,h) &= \begin{cases} 1 & \text{if } c = c_2 \text{ and } h \text{ ends in } c_1, w \text{ for some } w \\ 0 & \text{otherwise} \end{cases} \quad \forall\ c_1, c_2 \in V_C \\
g_{c_1,w}(c,h) &= \begin{cases} 1 & \text{if } c = c_1 \text{ and } h \text{ ends in } w \\ 0 & \text{otherwise} \end{cases} \quad \forall\ c_1 \in V_C \text{ and } w \in V_W
\end{aligned}$$

This description of the model still leaves open the question of what connectors were associated with the words in the training text. We parsed the Switchboard corpus using the robust link grammar parser and generated one linkage per sentence (rather than generating all of the linkages of the sentences). For each word we chose the connector on its longest left link if had a left link, otherwise we used the connector on the longest right link. Our preference of long links over short ones is because longer links seem to be more informative than shorter ones and some short range information is already captured by the features $f_{w_1,w_2}(w,h)$.

Table 2 lists the perplexities of a test corpus for three language models. The first model is a bigram model using a backoff scheme [24]. The second is our simple maximum entropy model using only the 100 most common connectors. The third is a maximum entropy model that ignores all connector information. It is included to aid in determining the benefit of connectors in the language model.

The perplexities of all three models are fairly close. Like many of the class-based n-gram models to which our simple model bears some resemblance, local part-of-speech-like information

topic	*high mutual information words*					
Air Pollution	pollution	air	cars	ozone	smog	smoke
	car	huh_uh	freon	acid	diesel	industry
Aids	aids	research	disease	cancer	spending	blood
	money	funding	who	died	spend	abstinence
Recycling	recycling	cans	recycle	paper	plastic	trash
	glass	aluminum	bottles	garbage	newspapers	recycled
Exercise and Fitness	exercise	aerobics	weight	walk	bike	walking
	shape	exercising	weights	doing	week	running

Table 3: Words with high mutual information with a given topic

does not seem to be sufficient to produce improved models. Possibly by allowing part-of-speech-like information to influence longer range dependencies between words might show some improvement. We examine this in section 7.2.1.

7.1.2. Topic dependent model

The conversations in the Switchboard corpus are on specific topics. We created lists, one for each topic, of the words in the training corpus with high mutual information with that topic. Table 3 contains some of the words on the lists for four topics. From examining just a few of the words with high mutual information with a topic, one can usually easily infer what the topic is.

For a given topic, we built a topic dependent model. This model was first trained using the same features and training data as the simple model. The model was then trained once again but this time, the training data was composed only of conversations with the given topic, and most of the parameters (λ_i) found earlier were fixed and not permitted to change. The only parameters allowed to change were those that corresponded to features $f_{w_1}(w,h)$ and $f_{w_1,c}(w,h)$ for which w_1 or w_1 with its connector c were in the topical training data. The choice of allowing the "unigram" parameters to change is clear: we want the model to account for the distribution of words within a given topic. We also allowed the parameters corresponding to $f_{w_1,c}(w,h)$ to change as sometimes words are only relevant to a topic if they are associated with certain connector types. For example, the distribution of the word "trash" when used as a verb within the "Recycling" topic is probably quite similar to its distribution in other topics. The distribution of "trash" used as a noun however, would be quite different. We built these models for the fifteen topics covered by our test corpus. When tested against data in the corresponding topics, 80% of the models produced lower perplexity and 20% higher perplexity.[3] These results are presented in table 4 where we also include information on the number of conversations and their sizes in the training and test data. It is interesting to note that the topic dependent models have higher perplexities only when there is limited training data. Additional experiments need to be performed to determine if there is a true relationship between the size of the training data and the relative performance of topic dependent versus topic independent models.

A reduction in perplexity usually occurs because the parameters associated with topic dependent words are increased over their counterparts in the topic independent model. For example, table 5 compares the unigram parameters for the high mutual information words in the "Air Pollution"

[3]We suspect that the maximum entropy models might have been overtrained.

topic	*training data*		*test data*		*perplexity*	
	#conv.	*avg. size*	*#conv.*	*avg. size*	*topic indep.*	*topic dep.*
Air pollution	30	1504.47	2	2346.00	126.857	115.375 (-9.05%)
Music	45	1428.24	2	1822.00	93.952	82.568 (-12.12%)
Crime	42	1490.55	2	2211.50	97.749	92.810 (-5.05%)
Gun control	37	1486.62	2	2033.50	101.300	90.708 (-10.46%)
Buying a car	53	1466.49	2	2336.50	95.701	85.584 (-10.57%)
Universal public service	28	1619.93	2	1612.00	94.190	89.967 (-4.48%)
Latin America	9	1729.89	1	1699.00	149.936	152.422 (+3.00%)
Painting	10	1598.00	1	1328.00	88.209	80.137 (-9.15%)
Pets	42	1607.02	2	1740.00	101.634	90.275 (-11.18%)
Public education	50	1433.80	2	1674.50	85.472	82.815 (-3.11%)
Exercise and fitness	46	1527.37	2	1990.00	109.964	98.244 (-10.66%)
Family life	47	1503.13	2	2009.00	90.988	90.375 (-0.67%)
Ethics in government	8	1098.25	1	1075.00	96.005	97.685 (+1.75%)
Space flight & exploration	8	1194.37	1	1339.00	90.447	93.811 (+3.72%)
Woodworking	7	1137.00	1	1334.00	111.526	108.617 (-2.60%)

Table 4: Comparison of perplexities for topic independent and topic dependent models

word	*topic dependent*	*topic independent*
pollution	3.250	0.447
air	7.975	1.016
cars	5.118	1.191
ozone	0.969	0.087
smog	0.708	0.065
smoke	1.814	0.257
car	6.148	2.452
huh_uh	2.641	1.023
freon	0.765	0.077
acid	0.508	0.057
diesel	1.105	0.116
industry	1.584	0.301

Table 5: Comparison of parameters for “Air Pollution” and topic independent models

topic dependent model with the same words in the topic independent model. The parameters are remarkably different, with much higher values corresponding to the topic dependent model. The connector information also slightly helped the topic dependent models; in all cases but one, topic dependent models with no connector information had (slightly) higher perplexity than ones with connector information. Models similar to these "connectorless" models were used in a speech recognition system in [29] with mixed results. In the next section we will see how we can try to exploit this relationship between topic and connector information.

Not surprisingly, when examining the perplexity of the model on test data at sentence intervals, we found that the perplexity decreases when the conversation is on topic and increases as the topic drifts.

While the topic dependent experiments were a little artificial in the sense that the topic dependent models were tested against data hand-chosen to correspond with the topic, topic detection for the Switchboard corpus will probably be quite easy. A general language model could be built that would use a topic independent component until the topic became clear at which time it would switch to a topic dependent component.

As part of our future work we describe a model in which the topic dependent nature of the Switchboard corpus can be brought to bear without having to resort to topic detection.

7.2. Future work

In this section we discuss some of the work we intend to perform as part of this thesis.

7.2.1. Finite-state approximations

Trigger model

Triggers have great potential in domains that are highly topic dependent. In such domains, the highly correlated words found (using mutual information or some other technique) should mainly be those that pertain specifically to the same topic. When a trigger word appears in a document, it gives some indication of the document's topic. The probability the model gives to the triggered words will increase, causing the model to give greater preference to words relating to the topic.

We can try to improve the utility of triggers by bringing grammatical considerations to bear. Some words might be related to a conversation's topic (and thus should trigger some topic dependent words or be triggered by some topic dependent words) only when used in certain ways. One example of this we previously saw: "trash" used a noun is certainly related to the topic "Recycling" but when used as a verb it probably is not. We would not want an instance of "trash" to incorrectly trigger words when it is not used in connection to recycling.

Using information gathered from parsing the Switchboard corpus with the robust link grammar parser, we are currently building trigger models to help us discover the extent to which triggers help in reducing perplexity in the Switchboard domain and the degree to which the simple grammatical considerations mentioned above will help or hinder us. We are close to completing these experiments.

Training using more linkages

In the work presented up to now, we only used information from a single parse of each sentence of the corpus when in fact, the robust parser produces many additional parses, averaging 114 per sentence on the Switchboard corpus. There are a number of other issues such as coordination phenomena for which our language models took the "easy way out" and chose to mainly ignore some of the difficulties involved. We plan on incorporating data from all of the parses and to take matters such as coordination phenomena into consideration in our future models. We have completed some preliminary work towards this goal.

Other models

Some of our future work will involve developing more sophisticated language models by using data produced by our robust parser. In the first stages of this work we might continue to use finite-state approximations to stochastic grammars. We will use the GMM framework and it will allow us evaluate the benefits of adding or deleting various language components.

There are a number of areas to explore within this framework including the use of longer range grammatical structures such as subject/verb agreement as information sources. The robust link grammar parser will allow us to capture many grammatical relations and we expect that a number of them will be beneficial to language modeling.

7.2.2. Feature induction

In [40] a field induction algorithm is given to incrementally construct random fields and it is applied to discover morphological features of words. We plan on exploring the use of this algorithm to judge the relative worth of features and to incrementally build language models.

7.2.3. Stochastic grammars

We would like to design a probabilistic model of link grammars in the same general vein as [28] and use it on the Switchboard corpus. We feel that this will be the ultimate test of the our abilities to build better language models by incorporating grammatical information. We must remember, however, not to lose sight of the benefits of local lexical information and to include them in our language model. This can be done at the grammar level by including n-gram disjuncts, but we will also try to accomplish this at the level of language model design.

7.2.4. Treebank parses

One of the problems with training our models using link grammar parses is that we aren't guaranteed to have the correct parses of corpus sentences. Most of the parses produced probably have at least some small inaccuracies. If the Switchboard corpus is treebanked, we will consider using the treebank parses for training.

7.2.5. Lattice parsing

Another potentially useful change to the link grammar parsing algorithm would be to have it efficiently parse word lattices. Speech recognition systems generate lattices of hypotheses and efficient examination of the lattices by language models could be extremely beneficial.

7.2.6. System integration

While language models using grammatical constructs that demonstrate improvements as determined by reductions in perplexity are a good objective, it would be even better if we could use the language models to improve existing speech recognition systems. We hope to accomplish this by integrating language models into one of the later stages of a speech recognition system at Carnegie Mellon.

7.3. Possible future work

In this section we discuss some of the work that we don't expect to complete for this thesis but we might explore depending on the directions our research takes.

7.3.1. Prefix parser

For speech recognition tasks it is often useful to have a language model that is able to compute probabilities as it examines the sentence from left to right. Many of the language models today have this property. As the link grammar parsing algorithm does not readily lend itself to left to right parsing, link grammar based probabilistic language models will not either. One possible direction for work is to design and implement an efficient link grammar parser that works from left to right.

7.3.2. Early system integration

Earlier we proposed the integration of a language model with a speech recognition system. Unfortunately this integration will probably be at an advanced stage in the recognition process where it is easier to perform but where the benefits will be fewer. One possible direction of research would be to explore the integration of the language model into earlier stages of the recognition process.

7.3.3. Use in a speech understanding system

While we probably won't work on this, it might be useful to try and integrate a link grammar based language model with a speech understanding system. Some of these systems try to understand the speech only after it has been transcribed, separating the tasks of recognition and understanding into two distinct units. We feel that language models that utilize grammatical structure might help integrate these two areas together, with the understanding task helping to guide the recognition task.

7.3.4. Grammar inference

Our robust parser currently handles unknown words by assigning them some common disjuncts shared by many words. In some sentences, the contexts surrounding unknown words are sufficient to enforce their correct usages but other times they are not. One possible research direction would be to explore how to infer grammar for unknown words and how to fine-tune and possibly correct grammar for existing words.

8. Contributions

Some of the expected contributions of this work are:

1. A robust parsing algorithm for link grammars. This parser will be extremely adept at finding useful structure, even on highly ungrammatical text such as found in the Switchboard corpus.
2. An version of the parser that can be efficiently used on word lattices.
3. Demonstrating the feasibility and utility of maximum entropy models with features that are dependent on hidden data.
4. The first successful large-scale attempt at using a language model that utilizes grammatical structure to reduce perplexity.
5. Use of a language model that utilizes grammatical structure to reduce the error rate of a speech recognition system.
6. Learning what kinds of grammar-related features are useful in language modeling. A completely objective result could be given if we are successful in inducing "good" features automatically. If not, we still expect to be able to give a subjective view of the utility of various features in language modeling.

9. Schedule

Some of the milestones in the thesis schedule include:

Oct 31, 1995 complete trigger model.

Dec 15, 1996 complete training models using more linkages.

Jan 31, 1996 complete implementation of new models within the finite-state framework.

Feb 31, 1996 have some results on the use of these models for speech recognition.

May 31, 1996 preliminary results on stochastic grammar framework.

Nov 31, 1996 have some results on the use of these models for speech recognition.

Jan 31, 1997 Results on lattice parsing.

Apr 1, 1997 Defend.

10. Acknowledgements

This represents joint work with John Lafferty and Daniel Sleator. I am grateful to them for their significant participation in this work. I would also like to thank Sharyn Henry and James Landay for their useful comments on preliminary versions of this document.

References

[1] L. R. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, March 1983.

[2] J. K. Baker. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, 1979.

[3] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 1995. To appear.

[4] E. Black, F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of DARPA Speech and Natural Language Workshop*, Feb 1992.

[5] E. Black, F. Jelinek, J. Lafferty, and S. Roukos. Decision tree models applied to the labeling of text with parts-of-speech. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 117–121, Feb 1992.

[6] P. Brown, V. Della Pietra, P. de Souza, J. Lai, and R. Mercer. Class based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[7] P. E. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, Mar 1992.

[8] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[9] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[10] N. Chomsky. Formal properties of grammars. In L. D. Luce, R. R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, pages 323–418. Wiley, New York, 1963.

[11] K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 695–698, 1989.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[13] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[15] J. Godfrey, E. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research development. In *Proc. ICASSP-92*, pages I–517–520, 1992.

[16] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3, 4):237–264, Dec 1953.

[17] D. Grinberg, J. Lafferty, and D. Sleator. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague/Karlovy Vary, Czech Republic, 1995. Also issued as Technical Report CMU-CS-95-125, School of Computer Science, Carnegie Mellon University, 1995.

[18] E. T. Jaynes. Brandeis lectures (1963). In R. D. Rosenkrantz, editor, *Papers on Probability, Statistics and Statistical Physics*, pages 39–76. D. Reidel Publishing Company, Dordrecht, Holland, 1983.

[19] F. Jelinek. Self-organizing language models for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann Publishers, Inc., 1990.

[20] F. Jelinek and J. D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323, 1991.

[21] F. Jelinek, J. D. Lafferty, and R. L. Mercer. Basic methods of probabilistic context free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding: Recent Advances*, pages 345–360. Springer-Verlag, 1992. NATO ASI Series, Vol. F75.

[22] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. North–Holland Publishing Company, 1980.

[23] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. A dynamic language model for speech recognition. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 293–295, Feb 1991.

[24] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401, 1987.

[25] R. Kuhn and R. De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(6):570–583, June 1990.

[26] R. Kuhn and R. De Mori. Correction to a cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(6):691–692, June 1992.

[27] J. Lafferty. Gibbs-Markov models. In *Computing Science and Statistics: Proceedings of the 27th Symposium on the Interface*. Interface Foundation, 1995.

[28] J. Lafferty, D. Sleator, and D. Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, Cambridge, MA, October 1992. Also issued as Technical Report CMU-CS-92-181, School of Computer Science, Carnegie Mellon University, 1992.

[29] J. Lafferty and B. Suhm. Cluster expansions and iterative scaling of maximum entropy language models. In *Proceedings of the Fifteenth International Workshop on Maximum Entropy and Bayesian Methods*. Kluwer Academic Publishers, 1995.

[30] A. Lavie and M. Tomita. GLR*: An efficient noise-skipping parsing algorithm for context free grammars. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 123–134, 1993.

[31] K. J. Lee, C. J. Kweon, J. Seo, and G. C. Kim. A robust parser based on syntactic information. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, 1995.

[32] G. Lyon. Syntax-directed least-errors analysis for context-free languages. *Communications of the ACM*, 17(1):3–14, 1974.

[33] D. M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, Feb 1994.

[34] D. M. Magerman and M. P. Marcus. Pearl: A probabilistic chart parser. In *Proceedings of the European ACL Conference*, Berlin, Germany, Mar 1991.

[35] D. M. Magerman and C. Weir. Efficiency, robustness, and accuracy in Picky chart parsing. In *Proceedings of the Association for Computational Linguistics*, pages 40–47, Newark, Delaware, 1992.

[36] K. Mark, M. Miller, U. Grenander, and S. Abney. Parameter estimation for constrained context-free language models. In *Proceedings of the Speech and Natural Language Workshop*, pages 146–149, Harriman, New York, 1992.

[37] C. S. Mellish. Some chart-based techniques for parsing ill-formed input. In *Proceedings of the Association for Computational Linguistics*, pages 102–109, 1989.

[38] M. I. Miller and J. A. O'sullivan. Entropies and combinatorics of random branching processes and context-free languages. *IEEE Transactions on Information Theory*, 38(4), Mar 1992.

[39] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures. Inference and estimation of a long-range trigram model. In *Proceedings of the Second International Colloquium on Grammatical Inference and Applications*. Springer–Verlag, 1994. Lecture Notes in Artificial Intelligence, 862.

[40] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, 1995.

[41] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann Publishers, Inc., 1990.

[42] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, Carnegie Mellon University, Apr 1994. Available as Technical Report CMU-CS-94-138.

[43] C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.

[44] D. D. K. Sleator and D. Temperley. Parsing English with a link grammar. Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, 1991.