Automatic Acquisition of Phrase Grammars for Stochastic Language Modeling

Giuseppe Riccardi and Srinivas Bangalore
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 09732
{dsp3,srini}@research.att.com

Abstract

Phrase-based language models have been recognized to have an advantage over word-based language models since they allow us to capture long spanning dependencies. Class based language models have been used to improve model generalization and overcome problems with data sparseness. In this paper, we present a novel approach for combining the phrase acquisition with class construction process to automatically acquire phrase-grammar fragments from a given corpus. The phrase-grammar learning is decomposed into two sub-problems, namely the phrase acquisition and feature selection. The phrase acquisition is based on entropy minimization and the feature selection is driven by the *entropy reduction principle*. We further demonstrate that the phrase-grammar based n-gram language model significantly outperforms a phrase-based n-gram language model in an end-to-end evaluation of a spoken language application.

1 Introduction

Traditionally, n-gram language models implicitly assume *words* as the basic lexical unit. However, certain word sequences (phrases) are recurrent in constrained domain languages and can be thought as a single lexical entry (e.g. `by and large`, `I would like to`, `United States of America`, etc..). A traditional word n-gram based language model can benefit greatly by using variable length units to capture long spanning dependencies, for any given order n of the model. Furthermore, language modeling based on longer length units is applicable to languages which do not have a predefined notion of a *word*. However, the problem of data sparseness is more acute in phrase-based language models than in word-based language models. Clustering words into classes has been used to overcome data sparseness in word-based language models (et.al., 1992; Kneser and Ney, 1993; Pereira et al., 1993; McCandless and Glass, 1993; Bellegarda et al., 1996; Saul and Pereira, 1997). Although the automatically acquired phrases can be later clustered into classes to overcome data sparseness, we present a novel approach of combining the construction of classes during the acquisition of phrases. This integration of phrase acquisition and class construction results in the acquisition of phrase-grammar fragments. In (Gorin, 1996; Arai et al., 1997), grammar fragment acquisition is performed through Kullback-Liebler divergence techniques with application to topic classification from text.

Although phrase-grammar fragments reduce the problem of data sparseness, they can result in over-generalization. For example, one of the classes induced in our experiments was `C1 = {and, but, because}` which one might call the class of conjunctions. However, this class was part of a phrase-grammar fragment such as `A T C1 T` which results in phrases `A T and T, A T but T, A T because T` - a clear case of over-generalization given our corpus. Hence we need to further stochastically separate phrases generated by a phrase-grammar fragment.

In this paper, we present our approach to integrating phrase acquisition and clustering and our technique to specialize the acquired phrase fragments. We extensively evaluate the effectiveness of phrase-grammar based n-gram language model and demonstrate that it outperforms a phrase-based n-gram language model in an end-to-end evaluation of a spoken language application. The outline of the paper is as follows. In Section 2, we review the phrase acquisition algorithm presented in (Riccardi et al., 1997). In Section 3, we discuss our approach to phrase acquisition and clustering respectively. The algorithm integrating the phrase acquisition and clustering processes is presented in Section 4. The spoken language application for automatic call routing (*How May I Help You?* (HMIHY)) that is used for evaluating our approach and the results of our experiments are described in Section 5.

2 Learning Phrases

In previous work, we have shown the effectiveness of incorporating manually selected phrases for re-

ducing the test set perplexity[1] and the word error rate of a large vocabulary recognizer (Riccardi et al., 1995; Riccardi et al., 1996). However, a critical issue for the design of a language model based on phrases is the algorithm that automatically chooses the units by optimizing a suitable cost function. For improving the prediction of word probabilities, the criterion we used is the minimization of the language perplexity $PP(\mathcal{T})$ on a training corpus $\mathcal{T}$. This algorithm for extracting phrases from a training corpus is similar in spirit to (Giachin, 1995), but differs in the language model components and optimization parameters (Riccardi et al., 1997). In addition, we extensively evaluate the effectiveness of phrase n-gram ($n \geq 2$) language models by means of an end-to-end evaluation of a spoken language system (see Section 5). The phrase acquisition method is a greedy algorithm that performs local optimization based on an iterative process which converges to a local minimum of $PP(\mathcal{T})$. As depicted in Figure 1, the algorithm consists of three main parts:

- Generation and ranking of a set of candidate phrases. This step is repeated at each iteration to constrain the search for all possible symbol sequences observed in the training corpus.
- Each candidate phrase is evaluated in terms of the training set perplexity.
- At the end of the iteration, the set of selected phrases is used to filter the training corpus and replace each occurrence of the phrase with a new lexical unit. The filtered training corpus will be referenced as $\mathcal{T}_f$.

In the first step of the procedure, a set of candidate phrases (unit pairs) [2] is drawn out of a training corpus $\mathcal{T}$ and ranked according to a correlation coefficient. The most used measure for the interdependence of two events is the mutual information $MI(x,y) = log\frac{P(x,y)}{P(x)P(y)}$. However, in this experiment, we use a correlation coefficient that has provided the best convergence speed for the optimization procedure:

$$\rho_{x,y} = \frac{P(x,y)}{P(x)+P(y)} \qquad (1)$$

where $P(x)$ is the probability of symbol x. The coefficient $\rho_{x,y}$ ($0 \leq \rho_{x,y} \leq 0.5$) is easily extended to define $\rho_{x_1,x_2,\ldots,x_n}$ for the n-tuple $(x_1,x_2,\ldots,x_n)$ ($0 \leq \rho_{x_1,x_2,\ldots,x_n} \leq 1/n$). Phrases (x,y) with high $\rho_{x,y}$ or $MI(x,y)$ are such that $P(x,y) \simeq P(x) \simeq P(y)$. In the case of $P(x,y) = P(x) = P(y)$, $\rho_{x,y} = 0.5$ while $MI = -logP(x)$. Namely, the ranking by MI is biased towards events with low probability events which are not likely to be selected by our Maximum Likelihood algorithm. In fact, the phrase (x,y) will be selected only if $P(x,y) \simeq P(x) \simeq P(y)$ *and* the training set perplexity is decreased when (x,y) is treated as a single unit. In Figure 2 we show the behavior of the training set perplexity (*learning* curve) by incorporating an increasing number of selected phrases using $\rho_{x,y}$ and $MI(x,y)$ as ranking coefficients. In particular, after evaluating 1000 phrases and selecting 300 of those, the perplexity decrease is 20% and 4% using $\rho_{x,y}$ and $MI(x,y)$ respectively. Each of the candidate phrases (x,y) is treated as a single unit in order to build a stochastic model λ of

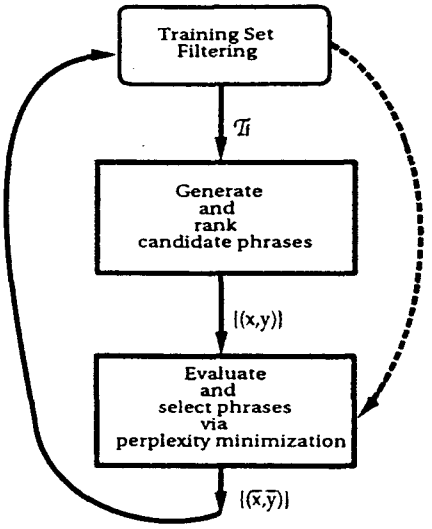

Figure 1: Algorithm for phrase selection

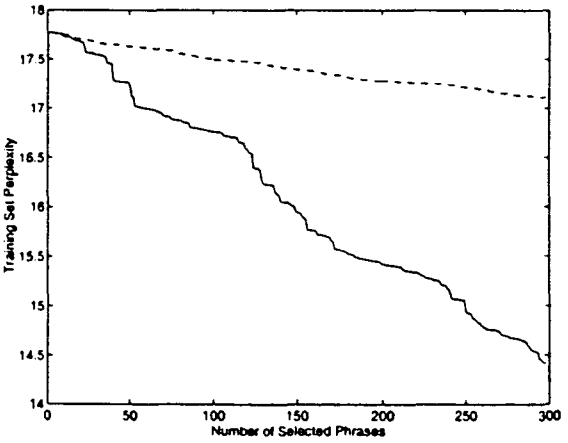

Figure 2: Training set perplexity *vs* number of selected phrases using ρ (solid line) and MI (dashed line).

[1] The perplexity $PP(\mathcal{T})$ of a corpus $\mathcal{T}$ is $PP(\mathcal{T}) = \exp(-\frac{1}{n}\log P(\mathcal{T}))$, where n is the number of words in $\mathcal{T}$.

[2] We ranked symbol pairs and increased the phrase length by successive iteration. An additional speed up to the algorithm could be gained by ranking symbol k-tuples ($k \geq 2$) at each iteration.

k-th ($k \geq 2$) order based on the filtered training corpus, $\mathcal{T}_f$ [3]. Then, (x, y) is selected by the algorithm if $PP_\lambda(\mathcal{T}) < \mathcal{PP}(\mathcal{T})$. At the end of each iteration the set $\{(\bar{x}, \bar{y})\}$ is selected and employed to filter the training corpus. The algorithm iterates until the perplexity decrease saturates or a specified number of phrases have been selected. [4]

The second issue in building a phrase-based language model is the training of large vocabulary stochastic finite state machines. In (Riccardi et al., 1996) we present a unified framework for learning stochastic finite state machines (Variable Ngram Stochastic Automata, VNSA) from a given corpus $\mathcal{T}$ for large vocabulary tasks. The stochastic finite state machine learning algorithm in (Riccardi et al., 1995) is designed in such a way that it can recognize any possible sequence of basic unit while

- minimizing the number of parameters (states and transitions).
- computing state transition probabilities based on word, phrase and class n-grams by implementing different back-off strategies.

For the word sequence $W = w_1, w_2, \ldots, w_N$, a standard word n-gram model provides the following probability decomposition:

$$P(W) = \prod_i P(w_i | w_{i-n+1}, \ldots, w_{i-1}) \qquad (2)$$

The phrase n-gram model maps from W into a bracketed sequence such as $[w_1]_{f_1}[w_2, w_3]_{f_2}, \ldots, [w_{N-2}, w_{N-1}, w_N]_{f_M}$. Then, the probability $P(W)$ can be computed as:

$$P(W) = \prod_i P(f_i | f_{i-n+1}, \ldots, f_{i-1}) \qquad (3)$$

By comparing equations 2 and 3 it is evident how the phrase n-gram model allows for an increased right and left context in computing $P(W)$.

In order to evaluate the test perplexity performance of our phrase-based VNSA, we have split the *How May I Help You?* data collection into an 8K and 1K training and test set, respectively. In Figure 3, the test set perplexity is measured versus the VNSA orders for word and phrase language models. It is worth noticing that the largest perplexity decrease comes from using phrase bigram when compared against word bigram. Furthermore, the perplexity of the phrase models is always lower than the corresponding word models.

[3] At the first iteration $\mathcal{T} \equiv \mathcal{T}_f$.

[4] The language model estimation component of the algorithm guards against the problem of overfitting (Riccardi et al., 1996).

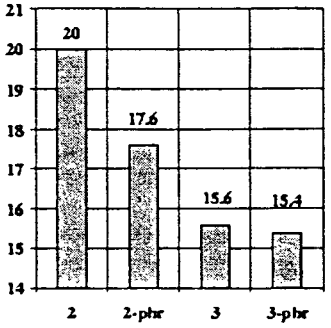

Figure 3: Test set perplexity *vs* VNSA Language Model Order

3 Clustering Phrases

In the context of language modeling, clustering has typically been used on *words* to induce classes that are then used to predict smoothed probabilities of occurrence for rare or unseen events in the training corpus. Most clustering schemes (et.al., 1992; Kneser and Ney, 1993; Pereira et al., 1993; McCandless and Glass, 1993; Bellegarda et al., 1996; Saul and Pereira, 1997) use the average entropy reduction to decide when two words fall into the same cluster. In contrast, our approach to clustering words is similar to Schutze(1992). The words to be clustered are each represented as a feature vector and similarity between two words is measured in terms of the distance between their feature vectors. Using these distances, words are clustered to produce a hierarchy. The hierarchy is then cut at a certain depth to produce clusters which are then ranked by a goodness metric. This method assigns each word to a unique class, thus producing hard clusters.

3.1 Vector Representation

A set of 50 high frequency words from the given corpus are designated as the "context words". The idea is that the high frequency words will mostly be function words which serve as good discriminators for content words (certain content words appear only with certain function words).

Each word is associated with a feature vector whose components are as follows:

1. Left context: The coocurrence frequency of each of the context word appearing in a window of 3 words to the left of the current word is computed. This determines the distribution of the context words to the left of the current word within a window of 3 words.

2. Right context: Similarly, the distribution of the context words appearing within a window of 3 words to the right of the current word is computed. This leaves us with adjacent words sharing a lot of the surrounding context and hence might end up

Class Index	Compactness Value	Class Members
C363	0.131	make place
C118	0.180	eight eighty five four nine oh one seven six three two zero
C357	0.190	bill charge
C260	0.216	an and because but so when
C300	0.233	K O ok
C301	0.236	from please
C277	0.241	again here
C202	0.252	is it's
C204	0.263	different third
C77	0.268	number numbers
C275	0.272	need needed want wanted
C256	0.274	assistance directory information
C197	0.278	all before happened
C68	0.278	ninety sixty
C41	0.290	his our the their
C199	0.291	called dialed got have
C27	0.296	as by in no not now of or something that that's there whatever working
C327	0.296	I I'm I've
C48	0.299	canada england france germany israel italy japan london mexico paris
C69	0.308	back direct out through
C143	0.312	connected going it
C89	0.314	arizona california carolina florida georgia illinois island jersey maryland michigan missouri ohio pennsylvania virginia west york
C23	0.323	be either go see somebody them
C90	0.332	about me off some up you

Table 1: The results of clustering words from the *How May I Help You ?* corpus

in the same class.[5] To prevent this situation from happening, we include two additional sets of features for the immediate left and immediate right contexts. Adjacent words then will have different immediate context profiles.

3. Immediate Left context: The distribution of the context words appearing to the immediate left of the current word.

4: Immediate Right context: The distribution of the context words appearing to the immediate right of the current word.

Thus each word of the vocabulary is represented by a 200 component vector. The frequencies of the components of the vector are normalized by the frequency of the word itself.

The Left and Right features are intended to capture the effects of wider range contexts thus collapsing contexts that differ only due to modifiers, while the Immediate Left and Right features are intended to capture the effects of local contexts. By including both sets of features, the effects of the local contexts are weighted more than the effects of the wider range contexts, a desirable property. The same result might be obtained by weighting the contributions of individual context positions differently, with the closest position weighted most heavily.

3.2 Distance Computation and Hierarchical clustering

Having set up a feature vector for each word, the similarity between two words is measured using the Manhattan distance metric between their feature vectors. Manhattan distance is based on the sum of the absolute value of the differences among the coordinates. This metric is much less sensitive to outliers than the Euclidean metric. We experimented with other distance metrics such as Euclidean and maximum, but Manhattan gave us the best results.

Having computed the distance matrix, the words are hierarchically clustered with a compact linkage[6], in which the distance between two clusters is the largest distance between a point in one cluster and a point in the other cluster(Jain and Dubes, 1988). A hierarchical clustering method was chosen since we expected to use the hierarchy as a back-off model. Also, since we don't know a priori the number of clusters we want, we did not use clustering schemes such as k-means clustering method where we would have to specify the number of clusters from the start.

[5] It is unlikely that with fine grained classes, two words belonging to the same class will follow each other.

[6] We tried other linkage strategies such as average linkage and connected linkage, but compact linkage gave the best results.

Class Index	Compactness Value	Class Members
D365	0.226	wrong:C77 second
D325	0.232	C256:C256 C256
D380	0.239	area:code:C118:C118:C118:C118:C118 C68
D386	0.243	a:C77 this:C77
D382	0.276	C260:C357:C143:to:another C260:C357:C143:to:my:home
D288	0.281	C327:C275:to:C363 I'd:like:to:C363 to:C363 yes:I'd:like:to:C363
D186	0.288	good:morning yes:ma'am yes:operator hello hi ma'am may well
D148	0.315	problems trouble
D87	0.315	A:T:C260:T C260:C327 C27:C27 C41:C77 C118 C143 C260 C197 C199 C202 C23 C260 C27 C277 C301 C69 C77 C90 operator to
D183	0.321	C118:C118:hundred C204 telephone
D143	0.326	new:C89 C48 C89 colorado massachusetts tennessee texas
D387	0.327	my:home my:home:phone
D4	0.336	my:calling my:calling:card my:card
D70	0.338	C199:a:wrong:C77 misdialed
D383	0.341	like:to:C363 trying:to:C363 would:like:to:C363
D381	0.347	like:to:C363:a:collect:call:to like:to:C363:collect:call would:like:to:C363:a:collect:call would:like:to:C363:a:collect:call:to
D159	0.347	C118:C118 C118:C118:C118 C118:C118:C118:C118:C118:C118 C118:C118:C118:C118:C118:C118:C118 C118:C118:C118:C118:C118:C118:C118:C118 C118:C118:C118:C118:C118:C118:C118:C118:C118 C118:C118:C118:C118:C118:C118:C118:C118:C118:C118:C118 area:code:C118:C118:C118 C300

Table 2: The results of the first iteration of combining phrase acquistion and clustering from the HMIHY corpus. (Words in a phrase are separated by a ":". The members of Ci's are shown in Table 1)

3.2.1 Choosing the number of clusters

One of the most tricky issues in clustering is the choice of the number of clusters after the clustering is complete. Instead of predetermining the number of clusters to be fixed, we use the median of the distances between clusters merged at the successive stages as the cutoff and prune the hierarchy at the point where the cluster distance exceeds the cutoff value. Clusters are defined by the structure of the tree above the cutoff point. (Note that the cluster distance increases as we climb up the hierarchy).

3.2.2 Ranking the clusters

Once the clusters are formed, the goodness of the cluster is measured by its compactness value. The compactness value of a cluster is simply the average distance of the members of the cluster from the centroid of the cluster. The components of the centroid vector is computed as the component-wise average of the vector representations of each of the members of the cluster.

The method described above is general in that it can be used to either cluster words and phrases. Table 1 illustrates the result of clustering words and Table 2 illustrates the result of clustering phrases for the training data from our application domain.

For example, the first iteration of the algorithm clusters words and the result is shown in Table 1. Each word in the corpus is replaced by its class label. If the word is not a member of any class then it is left unchanged. This transformed corpus is input to the phrase acquisition process. Figure 4 shows interesting and long phrases that are formed after the phrase acquisition process. Table 2 shows the result of subsequent clustering of the phrase-annotated corpus.

1. like:to:C363:a:collect:call:to
2. like:to:C363:collect:call
3. would:like:to:C363:a:collect:call
4. would:like:to:C363:a:collect:call:to

Figure 4: Sample phrases that include class label C363={make place}.The components of a phrase are separated by a :.

4 Learning Phrase Grammar

In the previous sections we have shown algorithms for acquiring (see section 2) and clustering (see section 3) phrases. While it is straightforward to pipeline the phrase acquisition and clustering algorithms, in the context of learning phrase-grammars they are not separable. Thus, we cannot first learn phrases and then cluster them or vice versa. For example, in order to cluster together the phrase `cut off` and `disconnected`, we first have to learn the phrase `cut off`. On the other hand, in order to learn the phrase `area code for <city>` we first have to learn the cluster `<city>`, containing city names (e.g. Boston, New York, etc..).

Learning phrase grammars can be thought as an iterative process that is composed of two language acquisition strategies. The goal is to search those features f, sequence of terminal and non-terminal symbols, that provide the highest learning rate (the entropy reduction within a learning interval, *first* strategy) *and* minimize the language entropy (*second* strategy, same as in section 2).

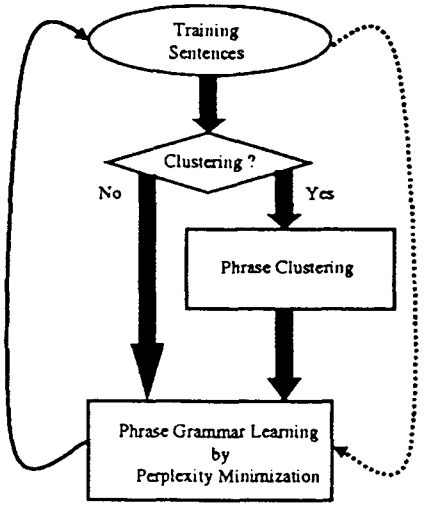

Figure 5: Algorithm for Phrase-grammar acquisition

Initially, the set of features f drawn from a corpus $\mathcal{T}$ contains terminal symbols V_0 only. New features can be generated by either

1. grouping (*conjunction* operator) an existing set of symbols, V_i, into phrases

or

2. map an existing set of symbols V_i into a new set of symbols V_{i+1} (*disjunction* operator) through the categorization provided by the clustering algorithm.

The whole symbol space is then given by $\mathcal{V} = \bigcup_i V_i$ as shown in Figure 6 and the problem of learning the *best* feature set is then decomposed into two subproblems: to find the *optimal* subset of $\mathcal{V}$ (*first learning strategy*) that gives us the *best* features (*second learning strategy*) generated by a given set V_i.

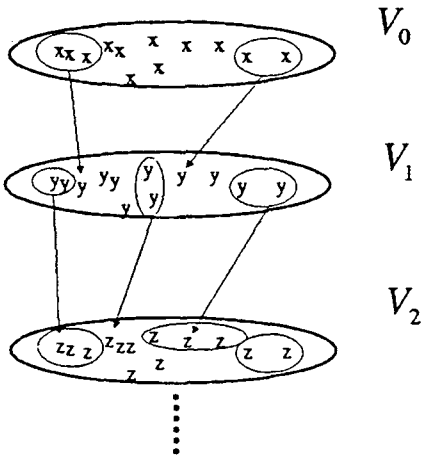

Figure 6: The sequence of symbol sets V_i generated by successive clustering steps.

In order to combine the two optimization problems, we have integrated them into a greedy algorithm as shown in Figure 5. In each algorithm iteration we *might* first cluster the current set of phrases and extract a set of non-terminal symbols and then acquire the phrases (containing terminal and non-terminal symbols) in order to minimize the language entropy. We use the clustering step of our algorithm to control the steepness of the learning curve within a subset V_i of the whole feature space. In fact, by varying the clustering rate (number of times clustering is performed for an entire acquisition experiment) we optimize the reduction of the language entropy for each feature selection (*entropy reduction principle*). Thus, the search for the optimal subset of $\mathcal{V}$ is designed so as to maximize the entropy reduction $\Delta H(\mathbf{f})$ over a set of features $\mathbf{f_i} = \{f_1, f_2, \ldots, f_m\}$ in V_i:

$$max_{\mathbf{f_i}} \Delta H(\mathbf{f_i}) = max_{\mathbf{f_i}} \hat{H}_{\lambda_o}(\mathcal{T}) - \hat{H}_{\lambda_{\mathbf{f_i}}}(\mathcal{T}) \quad (4)$$

$$= max_{\mathbf{f_i}} \frac{1}{n} log \frac{P_{\lambda_{\mathbf{f_i}}}(\mathcal{T})}{P_{\lambda_o}(\mathcal{T})} \quad (5)$$

where $\hat{H}_{\lambda_{\mathbf{f_i}}}(\mathcal{T})$ is the entropy of the corpus $\mathcal{T}$ based on the phrase n-gram model $\lambda_{\mathbf{f_i}}$ and λ_o is the initial model and equation 5 follows from equation 4 in the sense of the law of large numbers. The search space over all possible features $\mathbf{f}$ in equation 4 is built upon the notion of phrase ranking according to the ρ measure (see Section 2) and phrase clustering rate. By varying these two parameters we can search for the best learning strategies following the greedy algorithm given in Section 2. In Figure 7,

we give an example of *slow* and *quick* learning, defined by the rate of entropy reduction within an interval. The discontinuities in the learning curves correspond to the clustering algorithm step. The maximization in equation 5 is carried out for each interval between the entropy discontinuities. Therefore, the *quick* learning strategy provides the *best* learning curve in the sense of equation 5.

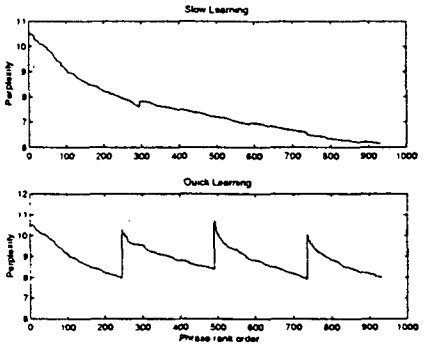

Figure 7: Examples of *slow* and *quick* phrase-grammar learning.

4.1 Training Language Models for Large Vocabulary Systems

Phrase-grammars allow for an increased generalization, since they can generate phrases that may never have been observed in the training corpus, but yet *similar* to the ones that have been observed. This generalization property is also used for smoothing the word probabilities in the context of stochastic language modeling for speech recognition and understanding. Standard class-based models smooth the word n-gram probability $P(w_i|w_{i-n+1},\ldots,w_{i-1})$ in the following way:

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = P(C_i|C_{i-n+1},\ldots,C_{i-1})P(w_i|C_i) \quad (6)$$

where $P(C_i|C_{i-n+1},\ldots,C_{i-1})$ is the class n-gram probability and $P(w_i|C_i)$ is the class membership probability. However, phrases recognized by the same phrase-grammar can actually occur within different syntactic contexts but their similarity is based on their most likely lexical context. In (Riccardi et al., 1996) we have developed a context dependent training algorithm of the phrase class probabilities. In particular,

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = P(C_i|C_{i-n+1},\ldots,C_{i-1};S)P(w_i|C_i;S) \quad (7)$$

where S is the state of the language model assigned by the VNSA model (Riccardi et al., 1996). In particular, $S = S(w_{i-n+1},\ldots,w_{i-1};\lambda_f)$ is determined by the word history $w_{i-n+1},\ldots,w_{i-1}$ and the phrase-grammar model λ_f. For example, our algorithm has acquired the conjunction cluster {but, and, because} that leads to generate phrases like A T and T or A T because T, the latter clearly an erroneous generalization given our corpus. However, training context dependent probabilities as shown in Equation 7 delivers a *stochastic* separation between the correct and incorrect phrases:

$$\log \frac{P(\texttt{A T and T})}{P(\texttt{A T but T})} = 5.7 \quad (8)$$

Given a set of phrases containing terminal and non terminal symbols, the goal of large vocabulary stochastic language modeling for speech recognition and understanding is to assign a probability to all terminal symbol sequences. One of the main motivation for learning phrase-grammars is to decrease the local uncertainty in decoding spontaneous speech by embedding tightly constrained structure in the large vocabulary automaton. The language models trained on the acquired phrase-grammars give a slight improvement in perplexity (average measure of uncertainty). Another figure of merit in evaluating a stochastic language model is its local entropy ($-\sum_i P(s_i|s)logP(s_i|s)$) which is related to the notion of the branching factor of a language model state s. In Figure 8 we plot the local entropy histograms for word, phrase and phrase-grammar bigram stochastic models. The word bigram distribution reflects the sparseness of the word pair constraints. The phrase-grammar based language model delivers a local entropy distribution skewed in the range $[0-1]$ because of the tight constraints enforced by the phrase-grammars.

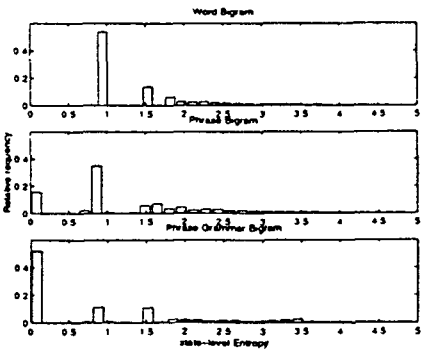

Figure 8: Local entropy histograms for word, phrase and phrase-grammar bigram VNSAs.

5 Spoken Language Application

We have applied the algorithms for phrase-grammar acquisition to the *How May I Help You?* (Gorin et al., 1997) speech understanding task. We briefly review the problem and the spoken language system.

The goal is to understand caller's responses to the open-ended prompt *How May I Help You?* and route such a call based on the meaning of the response. Thus we aim at extracting a relatively small number of semantic actions from the utterances of a *very large set* of users who are *not trained* to the system's capabilities and limitations.

The first utterance of each transaction has been transcribed and marked with a call-type by labelers. There are 14 call-types plus a class *other* for the complement class. In particular, we focused our study on the classification of the caller's first utterance in these dialogs. The spoken sentences vary widely in duration, with a distribution distinctively skewed around a mean value of 5.3 seconds corresponding to 19 words per utterance. Some examples of the first utterances are given below:

- `Yes ma'am where is area code two zero one?`
- `I'm tryn'a call and I can't get it to go through I wondered if you could try it for me please?`
- `Hello`

In the the training set there are $3.6K$ words which define the lexicon. The out-of-vocabulary (OOV) rate at the token level is 1.6%, yielding a sentence-level OOV rate of 30%. Significantly, only 50 out of the 100 lowest rank singletons were cities and names while the other were regular words like *authorized*, *realized*, etc.

For call type classification from speech we designed a large vocabulary one-step speech recognizer utilizing the phrase-grammar stochastic (section 4) model that achieved 60% word accuracy. Then, we categorized the decoded speech input into call-types, using the *salient* fragment classifier developed in (Gorin, 1996; Gorin et al., 1997). The *salient* phrases have the property of modeling local constraints of the language while carrying most of the semantic interpretation of the whole utterance. A block diagram of the speech understanding system is given in Figure 10. In an automated call router there are two important performance measures. The first is the probability of false rejection, where a call is falsely rejected or classified as *other*. Since such calls would be transferred to a human agent, this corresponds to a missed opportunity for automation. The second measure is the probability of correct classification. Errors in this dimension lead to misinterpretations that must be resolved by a dialog manager (Abella and Gorin, 1997). In Figure 9, we plot the probability of correct classification versus the probability of false rejection, for different speech recognition language models and the same classifier (Gorin et al., 1997). The curves are generated by varying a salience threshold (Gorin, 1996). In a dialog system, it would be useful even if the correct call-type was one of the top 2 choices of the decision rule (Abella and Gorin, 1997). Thus, in Figure 9 the classification scores are shown for the first and second ranked call-types identified by the understanding algorithm. Phrase-grammar trigram model is compared to the baseline system which is based on the phrase-based stochastic finite state machines described in (Gorin et al., 1997). The phrase-grammar model outperforms the baseline phrase-based model. and it achieves a 22% classification error rate reduction. The second set of curves (*Text*) in Figure 9 give an upper bound on the performance from speech experiments. It is worth noting, the rank 2 performance of the phrase-grammar model is aligned with rank 1 classification performance on the true transcriptions (dashed lines).

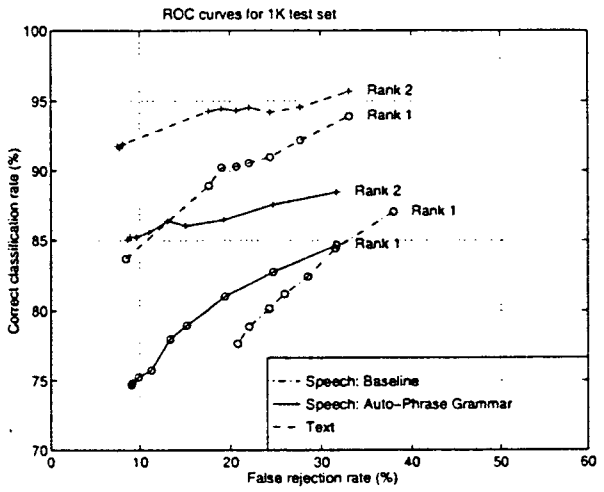

Figure 9: Rank 1 and 2 classification rate plot from text and speech with phrase-grammar trigram

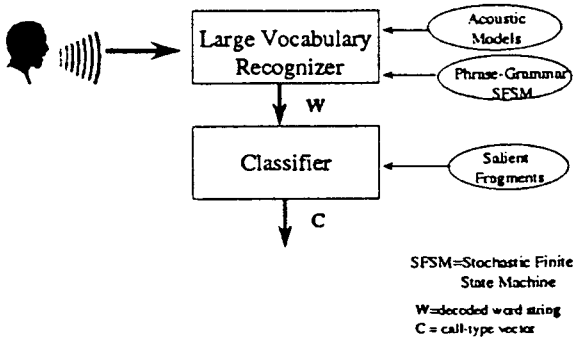

Figure 10: Block diagram of the speech understanding system

6 Conclusions

In this paper, we have presented a novel approach to automatically combine the acquisition of grammar fragments for language modeling. The phrase grammar learning is decomposed into two sub-problems, namely the phrase acquisition and feature selection. The phrase acquisition is based on entropy minimization and the feature selection is driven by the *entropy reduction principle.* This integration results in the learning of stochastic phrase-grammar fragments, which are then context dependent trained on the corpus at hand. We also demonstrated that a phrase-grammar based language model significantly outperformed a phrase-based language model in an end-to-end evaluation of a spoken language application.

References

A. Abella and A. L. Gorin. 1997. Generating semantically consistent inputs to a dialog manager. In *Proc. EUROSPEECH, European Conference on Speech Communication and Technology*, pages 1879–1882, September.

K. Arai, J. H. Wright, G. Riccardi, and A. L. Gorin. 1997. Grammar fragment acquisition using syntactic and semantic clustering. In *Proc. Workshop Spoken Language Understanding and Communication*, December.

J. Bellegarda, J. Butzberger, Y. Chow, N. Coccaro, and D. Naik. 1996. A novel word clustering algorithm based on latent semantic analysis. In *Proceedings of ICASSP-96*, pages 172–175.

Brown et.al. 1992. Class-based n-gram models of natural language. In *Computational Linguistics*, volume 18(4), pages 467–479.

E. Giachin. 1995. Phrase bigram for continous speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc.*, pages 225–228, May.

A. L. Gorin, G. Riccardi, and J. H. Wright. 1997. How May I Help You? In *Speech Communication*, volume 23, pages 113–127.

A. L. Gorin. 1996. Processing of semantic information in fluently spoken language. In *Proc. of Intl. Conf. on Spoken Language Processing*, October.

A.K. Jain and R.C. Dubes. 1988. *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice-Hall.

Reinhard Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Eurospeech.*

Michael K. McCandless and James R. Glass. 1993. Empirical acquisition of word and phrase classes in the atis domain. In *In Third European Conference on Speech Communication and Technology*, Berlin. Germany, September.

Fernando C. N. Pereira, Naftali Z. Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio.

G. Riccardi, E. Bocchieri, and R. Pieraccini. 1995. Non deterministic stochastic language models for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc.*, May.

G. Riccardi, R. Pieraccini, and E. Bocchieri. 1996. Stochastic automata for language modeling. In *Computer Speech and Language*, pages 265–293, December.

G. Riccardi, A. L. Gorin, A. Ljolje, and M. Riley. 1997. A spoken language understanding for automated call routing. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc.*, pages 1143–1146, April.

L. Saul and F. Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing.*

Hinrich Schutze. 1992. Dimensions of meaning. In *Proceedings of Supercomputing, Minneapolis MN.*, pages 787–796.