Information Extraction From Voicemail

Jing Huang and **Geoffrey Zweig** and **Mukund Padmanabhan**

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
USA
jhuang, gzweig, mukund@watson.ibm.com

Abstract

In this paper we address the problem of extracting key pieces of information from voicemail messages, such as the identity and phone number of the caller. This task differs from the named entity task in that the information we are interested in is a subset of the named entities in the message, and consequently, the need to pick the correct subset makes the problem more difficult. Also, the caller's identity may include information that is not typically associated with a named entity. In this work, we present three information extraction methods, one based on hand-crafted rules, one based on maximum entropy tagging, and one based on probabilistic transducer induction. We evaluate their performance on both manually transcribed messages and on the output of a speech recognition system.

1 Introduction

In recent years, the task of automatically extracting information from data has grown in importance, as a result of an increase in the number of publicly available archives and a realization of the commercial value of the available data. One aspect of information extraction (IE) is the retrieval of documents. Another aspect is that of identifying words from a stream of text that belong in predefined categories, for instance, "named entities" such as proper names, organizations, or numerics. Though most of the earlier IE work was done in the context of text sources, recently a great deal of work has also focused on extracting information from speech sources. Examples of this are the Spoken Document Retrieval (SDR) task (NIST, 1999), named entity (NE) extraction (DARPA, 1999; Miller et al., 2000; Kim and Woodland, 2000). The SDR task focused on Broadcast News and the NE task focused on both Broadcast News and telephone conversations.

In this paper, we focus on a source of conversational speech data, voicemail, that is found in relatively large volumes in the real-world, and that could benefit greatly from the use of IE techniques. The goal here is to query one's personal voicemail for items of information, without having to listen to the entire message. For instance, "who called today?", or "what is X's phone number?". Because of the importance of these key pieces of information, in this paper, we focus precisely on extracting the identity and the phone number of the caller. Other attempts at summarizing voicemail have been made in the past (Koumpis and Renals, 2000), however the goal there was to compress a voicemail message by summarizing it, and not to extract the answers to specific questions.

An interesting aspect of this research is that because a transcription of the voicemail is not available, speech recognition algorithms have to be used to convert the speech to text and the subsequent IE algorithms must operate on the transcription. One of the complications that we have to deal with is the fact that the state-of-the-art accuracy of speech recognition algorithms on this

type of data [1] is only in the neighborhood of 60-70% (Huang et al., 2000).

The task that is most similar to our work is named entity extraction from speech data (DARPA, 1999). Although the goal of the named entity task is similar - to identify the names of persons, locations, organizations, and temporal and numeric expressions - our task is different, and in some ways more difficult. There are two main reasons for this: first, caller and number information constitute a small fraction of all named entities. Not all person-names belong to callers, and not all digit strings specify phone-numbers. In this sense, the algorithms we use must be more precise than those for named entity detection. Second, the caller's identity may include information that is not typically found in a named entity, for example, "Joe on the third floor", rather than simply "Joe". We discuss our definitions of "caller" and "number" in Section 2.

To extract caller information from transcribed speech text, we implemented three different systems, spanning both statistical and non-statistical approaches. We evaluate these systems on manual voicemail transcriptions as well as the output of a speech recognizer. The first system is a simple rule-based system that uses trigger phrases to identify the information-bearing words. The second system is a maximum entropy model that tags the words in the transcription as belonging to one of the categories, "caller's identity", "phone number" or "other". The third system is a novel technique based on automatic stochastic-transducer induction. It aims to learn rules automatically from training data instead of requiring hand-crafted rules from experts. Although the results with this system are not yet as good as the other two, we consider it highly interesting because the technology is new and still open to significant advances.

The rest of the paper is organized as follows: Section 2 describes the database we are using; Section 3 contains a description of the baseline system; Section 4 describes the maximum entropy model and the associated features; Section 5 discusses the transducer induction technique; Section 6 contains our experimental results and Section 7 concludes our discussions.

[1]The large word error rate is due to the fact that the speech is spontaneous, and characterized by poor grammar, false starts, pauses, hesitations, etc. While this does not pose a problem for a human listener, it causes significant problems for speech recognition algorithms.

2 The Database

Our work focuses on a database of voicemail messages gathered at IBM, and made publicly available through the LDC. This database and related speech recognition work is described fully by (Huang et al., 2000). We worked with approximately $5,000$ messages, which we divided into $3,700$ messages for training, 500 for development test set, and 800 for evaluation test set. The messages were manually transcribed [2], and then a human tagger identified the portions of each message that specified the caller and any return numbers that were left. In this work, we take a broad view of what constitutes a caller or number. The caller was defined to be the consecutive sequence of words that best answered the question "who called?". The definition of a number we used is a sequence of consecutive words that enables a return call to be placed. Thus, for example, a caller might be "Angela *from P.C. Labs*," or "Peggy Cole *Reed Balla's secretary*". Similarly, a number may not be a digit string, for example: "*tieline* eight oh five six," or "*pager* one three five". No more than one caller was identified for a single message, though there could be multiple numbers. The training of the maximum entropy model and statistical transducer are done on these annotated scripts.

3 A Baseline Rule-Based System

In voicemail messages, people often identify themselves and give their phone numbers in highly stereotyped ways. So for example, someone might say, "Hi Joe it's Harry..." or "Give me a call back at extension one one eight four." Our baseline system takes advantage of this fact by enumerating a set of transduction rules - in the form of a *flex* program - that transduce out the key information in a call.

The baseline system is built around the notion of "trigger phrases". These hand-crafted phases are patterns that are used in the flex program to recognize caller's identity and phone numbers.

[2]The manual transcription has a 3% word error rate

Examples of trigger phrases are "Hi this is", and "Give me a call back at". In order to identify names and phone numbers as generally as possible, our baseline system has defined classes for person-names and numbers.

In addition to trigger phrases, "trigger suffixes" proved to be useful for identifying phone numbers. For example, the phrase "thanks bye" frequently occurs immediately after the caller's phone number. In general, a random sequence of digits cannot be labeled as a phone number; but, a sequence of digits followed by "thanks bye" is almost certainly the caller's phone number. So when the flex program matches a sequence of digits, it stores it; then it tries to match a trigger suffix. If this is successful, the digit string is recognized a phone number string. Otherwise the digit string is ignored.

Our baseline system has about 200 rules. Its creation was aided by an automatically generated list of short, commonly occurring phrases that were then manually scanned, generalized, and added to the flex program. It is the simplest of the systems presented, and achieves a good performance level, but suffers from the fact that a skilled person is required to identify the rules.

4 Maximum Entropy Model

Maximum entropy modeling is a powerful framework for constructing statistical models from data. It has been used in a variety of difficult classification tasks such as part-of-speech tagging (Ratnaparkhi, 1996), prepositional phrase attachment (Ratnaparkhi et al., 1994) and named entity tagging (Borthwick et al., 1998), and achieves state of the art performance. In the following, we briefly describe the application of these models to extracting caller's information from voicemail messages.

The problem of extracting the information pertaining to the callers identity and phone number can be thought of as a tagging problem, where the tags are "caller's identity," "caller's phone number" and "other." The objective is to tag each word in a message into one of these categories.

The information that can be used to predict a word's tag is the identity of the surrounding words and their associated tags. Let $\mathcal{H}$ denote the set of possible word and tag contexts, called "*histories*", and $\mathcal{T}$ denote the set of tags. The maxent model is then defined over $\mathcal{H} \times \mathcal{T}$,and predicts the conditional probability $p(t|h)$ for a tag t given the history h. The computation of this probability depends on a set of binary-valued "features" $f_i(h,t)$.

Given some training data and a set of features the maximum entropy estimation procedure computes a weight parameter α_i for every feature f_i and parameterizes $p(t|h)$ as follows:

$$p(t|h) = \frac{\prod_i \alpha_i^{f_i(h,t)}}{Z}$$

where Z is a normalization constant.

The role of the features is to identify characteristics in the histories that are strong predictors of specific tags. (for example, the tag "caller" is very often preceded by the word sequence "this is"). If a feature is a very strong predictor of a particular tag, then the corresponding α_i would be high. It is also possible that a particular feature may be a strong predictor of the absence of a particular tag, in which case the associated α_i would be near zero.

Training a maximum entropy model involves the selection of the features and the subsequent estimation of weight parameters α_i. The testing procedure involves a search to enumerate the candidate tag sequences for a message and choosing the one with highest probability. We use the "beam search" technique of (Ratnaparkhi, 1996) to search the space of all hypotheses.

4.1 Features

Designing effective features is crucial to the maxent model. In the following sections, we describe the various feature functions that we experimented with. We first preprocess the text in the following ways: (1) map rare words (with counts less than 5) to the symbol "UNKNOWN"; (2) map words in a name dictionary to the symbol "NAME." The first step is a way to handle out of vocabulary words in test data; the second step takes advantage of known names. This mapping makes the model focus on learning features which help to predict the location of the caller identity and leave the actual specific names later for extraction.

4.1.1 Unigram lexical features

To compute unigram lexical features, we used the neighboring two words, and the tags associated with the previous two words to define the history h_i as

$$h_i = w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}$$

The features are generated by scanning each pair (h_i, t_i) in the training data with feature template in Table 1. Note that although the window is two words on either side, the features are defined in terms of the value of a single word.

	Features		
$\forall w_i$	$w_i = X$	&	$t_i = T$
	$t_{i-1} = X$	&	$t_i = T$
	$t_{i-2}t_{i-1} = XY$	&	$t_i = T$
	$w_{i-1} = X$	&	$t_i = T$
	$w_{i-2} = X$	&	$t_i = T$
	$w_{i+1} = X$	&	$t_i = T$
	$w_{i+2} = X$	&	$t_i = T$

Table 1: Unigram features of the current history h_i.

4.1.2 Bigram lexical features

The trigger phrases used in the rule-based approach generally consist of several words, and turn out to be good predictors of the tags. In order to incorporate this information in the maximum entropy framework, we decided to use ngrams that occur in the surrounding word context to generate features. Due to data sparsity and computational cost, we restricted ourselves to using only bigrams. The bigram feature template is shown in Table 2.

	Features		
$\forall w_i$	$w_i = X$	&	$t_i = T$
	$t_{i-1} = X$	&	$t_i = T$
	$t_{i-2}t_{i-1} = XY$	&	$t_i = T$
	$w_{i-2}w_{i-1} = XY$	&	$t_i = T$
	$w_{i-1}w_i = XY$	&	$t_i = T$
	$w_i w_{i+1} = XY$	&	$t_i = T$
	$w_{i+1}w_{i+2} = XY$	&	$t_i = T$

Table 2: Bigram features of the current history h_i.

4.1.3 Dictionary features

First, a number dictionary is used to scan the training data and generate a code for each word which represents "number" or "other". Second, a multi-word dictionary is used to match known pre-caller trigger prefixes and after-phone-number trigger suffixes. The same code is assigned to each word in the matched string as either "pre-caller" or "after-phone-number". The combined stream of codes is added to the history h_i and used to generate features the same way the word sequence are used to generate lexical features.

4.2 Feature selection

In general, the feature templates define a very large number of features, and some method is needed to select only the most important ones. A simple way of doing this is to discard the features that are rarely seen in the data. Discarding all features with fewer than 10 occurrences resulted in about $10,000$ features. We also experimented with a more sophisticated incremental scheme. This procedure starts with no features and a uniform distribution $p(t|h)$, and sequentially adds the features that most increase the data likelihood. The procedure stops when the gain in likelihood on a cross-validation set becomes small.

5 Transducer Induction

Our baseline system is essentially a hand specified transducer, and in this section, we describe how such an item can be automatically induced from labeled training data. The overall goal is to take a set of labeled training examples in which the caller and number information has been tagged, and to learn a transducer such that when voicemail messages are used as input, the transducer emits only the information-bearing words. First we will present a brief description of how an automaton structure for voicemail messages can be learned from examples, and then we describe how to convert this to an appropriate transducer structure. Finally, we extend this process so that the training procedure acts hierarchically on different portions of the messages at different times.

In contrast to the baseline *flex* system, the transducers that we induce are nondeterministic and

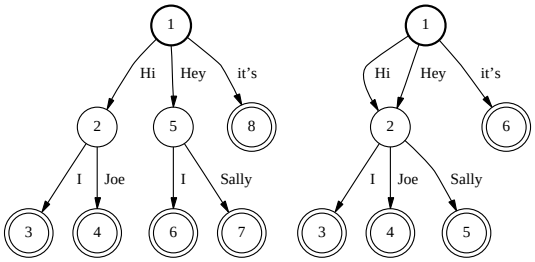

Figure 1: Graph structure before and after a merge.

stochastic – a given word sequence may align to multiple paths through the transducer. In the case that multiple alignments are possible, the lowest cost transduction is preferred, with the costs being determined by the transition probabilities encountered along the paths.

5.1 Inducing Finite State Automata

Many techniques have evolved for inducing finite state automata from word sequences, e.g. (Oncina and Vidal, 1993; Stolcke and Omohundro, 1994; Ron et al., 1998), and we chose to adapt the technique of (Ron et al., 1998). This is a simple method for inducing acyclic automata, and is attractive because of its simplicity and theoretical guarantees. Here we present only an abbreviated description of our implementation, and refer the reader to (Ron et al., 1998) for a full description of the original algorithm. In (Appelt and Martin, 1999), finite state transducers were also used for named entity extraction, but they were hand specified.

The basic idea of the structure induction algorithm is to start with a prefix tree, where arcs are labeled with words, that exactly represents all the word sequences in the training data, and then to gradually transform it, by merging internal states, into a directed acyclic graph that represents a generalization of the training data. An example of a merge operation is shown in Figure 1.

The decision to merge two nodes is based on the fact that a set of strings is rooted in each node of the tree, specified by the paths to all the reachable leaf nodes. A merge of two nodes is permissible when the corresponding sets of strings are statistically indistinguishable from one another. The precise definition of statistical similarity can be found in (Ron et al., 1998), and amounts to deeming two nodes indistinguishable unless one of them has a frequently occurring suffix that is rarely seen in the other. The exact ordering in which we merged nodes is a variant of the process described in (Ron et al., 1998) [3]. The transition probabilities are determined by aligning the training data to the induced automaton, and counting the number of times each arc is used.

5.2 Conversion to a Transducer

Once a structure is induced for the training data, it can be converted into an information extracting transducer in a straightforward manner. When the automaton is learned, we keep track of which words were found in information-bearing portions of the call, and which were not. The structure of the transducer is identical to that of the automaton, but each arc makes a transduction. If the arc is labeled with a word that was information-bearing in the training data, then the word itself is transduced out; otherwise, an <epsilon> is transduced.

5.3 Hierarchical Structure Induction

Conceptually, it is possible to induce a structure for voicemail messages in one step, using the algorithm described in the previous sections. In practice, we have found that this is a very difficult problem, and that it is expedient to break it into a number of simpler sub-problems. This has led us to develop a three-step induction process in which only short segments of text are processed at once.

First, all the examples of phone numbers are gathered together, and a structure is induced. Similarly, all the examples of caller's identities are collected, and a structure is induced for them To further simplify the task, we replaced number strings by the single symbol "NUMBER+", and person-names by the symbol "PERSON-NAME". The transition costs for these structures are estimated by aligning the training data, and counting

[3] A frontier of nodes is maintained, and is initialized to the children of the root. The weight of a node is defined as the number of strings rooted in it. At each step, the heaviest node is removed, and an attempt is made to merge it with another fronteir node, in order of decreasing weight. If a merge is possible, the result is placed on the frontier; otherwise, the heaviest node's children are added.

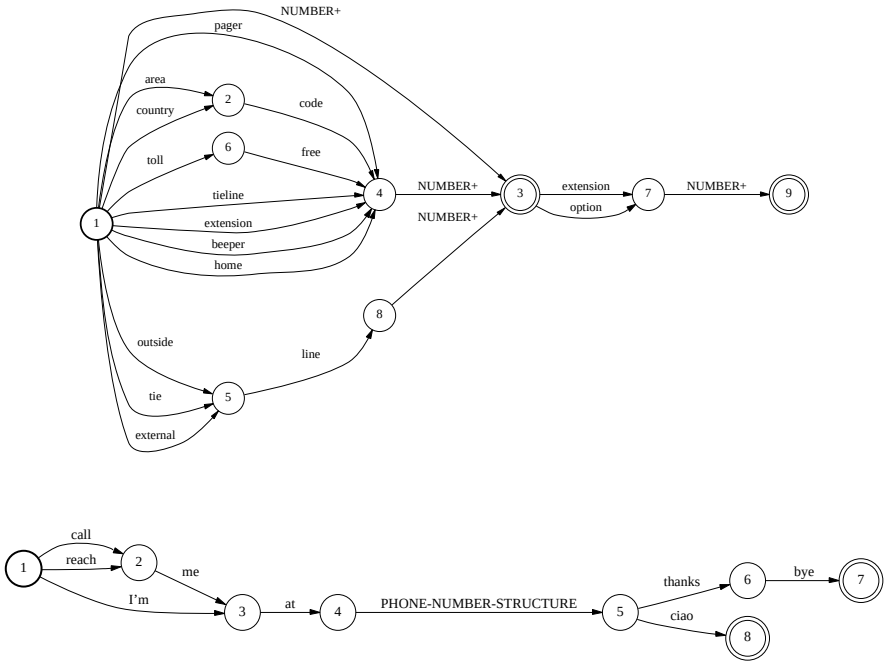

Figure 2: Induced structure for phone numbers (top), and a sub-graph of the second-level "number-segment" structure in which it is embedded (bottom). For clarity, transition probabilities are not displayed.

the number of times the different transitions out of each state are taken. A phone number structure induced in this way from a subset of the data is shown at the top of Figure 2.

In the second step, occurrences of names and numbers are replaced by single symbols, and the segments of text immediately surrounding them are extracted. This results in a database of examples like "Hi PERSON-NAME it's CALLER-STRUCTURE I wanted to ask you", or "call me at NUMBER-STRUCTURE thanks bye". In this example, the three words immediately preceding and following the number or caller are used. Using this database, a structure is induced for these segments of text, and the result is essentially an induced automaton that represents the trigger phrases that were manually identified in the baseline system. A small second level structure is shown at the bottom of Figure 2.

In the third step, the structure of a background language model is induced. The structures discovered in these three steps are then combined into a single large automaton that allows any sequence of caller, number, and background segments. For the system we used in our experiments, we used a unigram language model as the background. In the case that information-bearing patterns exist in the input, it is desirable for paths through the non-background portions of the final automaton to have a lower cost, and this is most likely with a high perplexity background model.

6 Experimental Results

To evaluate the performance of different systems, we use the conventional *precision*, *recall* and their *F-measure*. Significantly, we insist on **exact** matches for an answer to be counted as *correct*. The reason for this is that any error is liable to render the information useless, or detrimental. For example, an incorrect phone number can result in unwanted phone charges, and unpleasant conversations. This is different from typical named entity evaluation, where partial matches are given partial credit. Therefore, it should be understood that the precision and recall rates computed with this strict criterion *cannot* be compared to those from named entity detection tasks.

A summary of our results is presented in Tables

	P/C	R/C	F/C	P/N	R/N	F/N
baseline	73	68	70	81	83	82
ME1-U	88	75	81	90	78	84
ME1-B	89	80	84	88	78	83
ME2-U-f1	88	76	81	90	82	86
ME2-U-f12	87	78	82	90	83	86
ME2-B-f12	88	80	84	89	83	86
ME2-U-f12-I	87	78	82	89	81	85
ME2-B-f12-I	87	79	83	90	82	86
Transduction	21	43	29	52	78	63

Table 3: Precision and recall rates for different systems on manual voicemail transcriptions.

	P/C	R/C	F/C	P/N	R/N	F/N
baseline	22	17	19	52	54	53
ME2-U-f1	24	16	19	56	52	54

Table 4: Precision and recall rates for different systems on decoded voicemail messages.

3 and 4. Table 3 presents precision and recall rates when manual word transcriptions are used; Table 4 presents these numbers when speech recognition transcripts are used. On the heading line, P refers to precision, R to recall, F to F-measure, C to caller-identity, and N to phone number. Thus P/C denotes "precision on caller identity".

In these tables, the maximum entropy model is referred to as ME. ME1-U uses unigram lexical features only; ME1-B uses bigram lexical features only. ME1-B performs somewhat better than ME1-U, but uses more than double number of features.

ME2-U-f1 uses unigram lexical features and number dictionary features. It improves the recall of phone number by 4.5% upon ME1-U. ME2-U-f12 adds the trigger phrase dictionary features to ME2-U-f1, and it improves the recall of caller and phone numbers but degrades on the precision of both. Overall it improves a little on the F-meansures. ME2-B-f12 uses bigram lexical features, number dictionary features and trigger phrase dictionary features. It has the best recall of caller, again with over two times number of features of ME2-U-f12.

The above variants of ME features are chosen using simple count cutoff method. When the incremental feature selection is used, ME2-U-f12-I reduces the number of features from 9747 to 910 with minor performance loss; ME2-B-f12-I reduces the number of features from 25800 to 2125 with minor performance loss. This shows that the main power of the maxent model comes from a a very small subset of the possible features. Thus, if memory and speed are concerned, the incremental feature selection is highly recommended.

	P/C	R/C	F/C	P/N	R/N	F/N
baseline	66	66	66	71	72	71
ME2-U-f1	83	72	77	84	81	83

Table 5: Precision and recall rates for different systems on replaced decoded voicemail messages.

	P/C	R/C	F/C	P/N	R/N	F/N
baseline	77	36	49	85	76	80
ME2-U-f1	73	41	52	85	79	82

Table 6: Precision and recall of time-overlap for different systems on decoded voicemail messages.

There are several observations that can be made from these results. First, the maximum entropy approach systematically beats the baseline in terms of precision, and secondly it is better on recall of the caller's identity. We believe this is because the baseline has an imperfect set of rules for determining the end of a "caller identity" description. On the other hand, the baseline system has higher recall for phone numbers. The results of structure induction are worse than the other two methods, however as this is a novel approach in a developmental stage, we expect the performance will improve in the future.

Another important point is that there is a significant difference in performance between manual and decoded transcriptions. As expected, the precision and recall numbers are worse in the presence of transcription errors (the recognizer had a word error rate of about 35%). The degradation due to transcription errors could be caused by either: (i) corruption of words in the context surrounding the names and numbers; or (ii) corruption of the information itself. To investigate this, we did the following experiment: we replaced the regions of decoded text that correspond to the correct caller identity and phone number with the correct manual transcription, and redid the test.

The results are shown in Table 5. Compared to

the results on the manual transcription, the recall numbers for the maximum-entropy tagger are just slightly ($2-3\%$) worse, and precision is still high. This indicates that the corruption of the information content due to transcription errors is much more important than the corruption of the context.

If measured by the string error rate, none of our systems can be used to extract exact caller and phone number information directly from decoded voicemail. However, they can be used to locate the information in the message and highlight those positions. To evaluate the effectiveness of this approach, we computed precision and recall numbers in terms of the temporal overlap of the identified and true information bearing segments. Table 6 shows that the temporal location of phone numbers can be reliably determined, with an F-measure of 80%.

7 Conclusion

In this paper, we have developed several techniques for extracting key pieces of information from voicemail messages. In contrast to traditional named entity tasks, we are interested in identifying just a selected subset of the named entities that occur. We implemented and tested three methods on manual transcriptions and transcriptions generated by a speech recognition system. For a baseline, we used a *flex* program with a set of hand-specified information extraction rules. Two statistical systems are compared to the baseline, one based on maximum entropy modeling, and the other on transducer induction. Both the baseline and the maximum entropy model performed well on manually transcribed messages, while the structure induction still needs improvement. Although performance degrades significantly in the presence of speech racognition errors, it is still possible to reliably determine the sound segments corresponding to phone numbers.

References

Douglas E. Appelt and David Martin. 1999. Named entity extraction from speech: Approach and results using the textpro system. In *Proceedings of the DARPA Broadcast News Workshop* (DARPA, 1999).

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Nyu: Description of the mene named entity system as used in MUC-7. In *Seventh Message Understanding Conference(MUC-7)*. ARPA.

DARPA. 1999. *Proceedings of the DARPA Broadcast News Workshop*.

J. Huang, B. Kingsbury, L. Mangu, M. Padmanabhan, G. Saon, and G. Zweig. 2000. Recent improvements in speech recognition performance on large vocabulary conversational speech (voicemail and switchboard). In *Sixth International Conference on Spoken Language Processing*, Beijing, China.

Ji-Hwan Kim and P.C. Woodland. 2000. A rule-based named entity recognition system for speech input. In *Sixth International Conference on Spoken Language Processing*, Beijing, China.

Konstantinos Koumpis and Steve Renals. 2000. Transcription and summarization of voicemail speech. In *Sixth International Conference on Spoken Language Processing*, Beijing, China.

David Miller, Sean Boisen, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 2000. Named entity extraction from noisy input: Speech and ocr. In *Proceedings of ANLP-NAACL 2000*, pages 316–324.

NIST. 1999. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.

Jose Oncina and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the Human Language Technology Workshop*, pages 250–255, Plainsboro, N.J. ARPA.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part of Speech Tagger. In Eric Brill and Kenneth Church, editors, *Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, May 17–18.

Dana Ron, Yoram Singer, and Naftali Tishby. 1998. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, 56(2).

Andreas Stolcke and Stephen M. Omohundro. 1994. Best-first model merging for hidden markov model induction. Technical Report TR-94-003, International Computer Science Institute.