Information Extraction from Voicemail Transcripts

Martin Jansche
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
jansche.1@osu.edu

Steven P. Abney
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932, USA
abney@research.att.com

Abstract

Voicemail is not like email. Even such basic information as the name of the caller/ sender or a phone number for returning calls is not represented explicitly and must be obtained from message transcripts or other sources. We discuss techniques for doing this and the challenges these tasks present.

1 Introduction

When you're away from the phone and someone takes a message for you, at the very least you'd expect to be told who called and whether they left a number for you to call back. If the same call is picked up by a voicemail system, even such basic information like the name of the caller and their phone number may not be directly available, forcing one to listen to the entire message[1] in the worst case. By contrast, information about the sender of an email message has always been explicitly represented in the message headers, starting with early standardization attempts (Bhushan et al., 1973) and including the two decade old current standard (Crocker, 1982). Applications that aim to present voicemail messages through an email-like interface – take as an example the idea of a "uniform inbox" presentation of email, voicemail, and other kinds of messages[2] – must deal with the problem of how to obtain information analogous to what would be contained in email headers.

[1] The average message length in the corpus described below is 36 seconds.

[2] Similar issues arise with FAX messages, for example.

Here we will discuss one way of addressing this problem, treating it exclusively as the task of extracting relevant information from voicemail transcripts. In practice, e.g. in the context of a sophisticated voicemail front-end (Hirschberg et al., 2001) that is tightly integrated with an organization-wide voicemail system and private branch exchange (PBX), additional sources of information may be available: the voicemail system or the PBX might provide information about the originating station of a call, and speaker identification can be used to match a caller's voice against models of known callers (Rosenberg et al., 2001). Restricting our attention to voicemail transcripts means that our focus and goals are similar to those of Huang et al. (2001), but the features and techniques we use are very different.

While the present task may seem broadly similar to named entity extraction from broadcast news (Gotoh and Renals, 2000), it is quite distinct from the latter: first, we are only interested in a small subset of the named entities; and second, the structure of the voicemail transcripts in our corpus is very different from broadcast news and certain aspects of this structure can be exploited for extracting caller names.

Huang et al. (2001) discuss three approaches: hand-crafted rules; grammatical inference of subsequential transducers; and log-linear classifiers with bigram and trigram features used as taggers (Ratnaparkhi, 1996). While the latter are reported to yield the best overall performance, the hand-crafted rules resulted in higher recall. Our phone number extractor is based on a two-phase procedure that employs a small hand-crafted component to propose candidate phrases, followed by a classifier that retains the desirable candidates. This allows for more or less inde-

pendent optimization of recall and precision, somewhat similar to the PNrule classifier learner (Agarwal and Joshi, 2001; Joshi et al., 2001). We shall see that hand-crafted rules achieve very good recall, just as Huang et al. (2001) had observed, and the pruning phase successfully eliminates most undesirable candidates without affecting recall too much. Overall performance of our method is better than if we employ a log-linear model with trigram features.

The success of the method proposed here is also due to the use of a rich set of features for candidate classification. For example, the majority of phone numbers in voicemail messages has either four, seven, or ten digits, whereas nine digits would indicate a social security number. In our two-phase approach it is straightforward for the second-phase classifier to take the length of a candidate phone number into account. On the other hand, standard named entity taggers that use trigram features do not exploit this information, and doing so would entail significant changes to the underlying models and parameter estimation procedures.

The rest of this paper is organized as follows. A brief overview of the data we used in §2 is followed by a discussion of methods for extracting two kinds of caller information in §3. Methods for extracting telephone numbers are discussed in §4, and §5 summarizes and concludes.

2 Voicemail Corpus

Development and evaluation was done using a proprietary corpus of almost 10,000 voicemail messages that had been manually transcribed and marked up for content. Some more details about this corpus can be found in (Bacchiani, 2001). The relevant content labeling is perhaps best illustrated with an (anonymized) excerpt form a typical message transcript:

> ⟨greeting⟩ hi Jane ⟨/greeting⟩ ⟨caller⟩ this is Pat Caller ⟨/caller⟩ I just wanted to I know you've probably seen this or maybe you already know about it . . . so if you could give me a call at ⟨telno⟩ one two three four five ⟨/telno⟩ when you get the message I'd like to chat about it hope things are well with you ⟨closing⟩ talk to you soon ⟨/closing⟩

This transcript is representative of a large class of messages that start out with a short greeting followed by a phrase that identifies the caller either by name as above or by other means ('hi, it's me'). A phone number may be mentioned as part of the caller's self-identification, or is often mentioned near the end of the message. It may seem natural and obvious that voicemail messages should be structured in this way, and this prototypical structure can therefore be exploited for purposes of locating caller information or deciding whether a digit string constitutes a phone number. The next sections discuss this in more detail.

The corpus was partitioned into two subsets, with 8120 messages used for development and 1869 for evaluation. Approximately 5% of all messages are empty. Empty messages were not discarded from the evaluation set since they constitute realistic samples that the information extraction component has to cope with. The development set contains 7686 non-empty messages.

3 Caller Information

Of the non-empty messages in the development set, 7065 (92%) transcripts contain a marked-up caller phrase. Of those, 6731 messages mention a name in the caller phrase. Extracting caller information can be broken down into two slightly different tasks: we might want to reproduce the existing caller annotation as closely as possible, producing *caller phrases* like 'this is Pat Caller' or 'it's me'; or we might only be interested in *caller names* such as 'Pat Caller' in our above example. We make use of the fact that for the overwhelming majority of cases, the caller's self-identification occurs somewhere near the beginning of the message.

3.1 Caller Phrases

Most caller phrases tend to start one or two words into the message. This is because they are typically preceded by a one-word ('hi') or two-word ('hi Jane') greeting. Figure 1 shows the empirical distribution of the beginning of the caller phrase across the 7065 applicable transcripts in the development data. As can be seen, more than 97% of all caller phrases start somewhere between one and seven words from the beginning of the message,

though in one extreme case the start of the caller phrase occurred 135 words into the message.

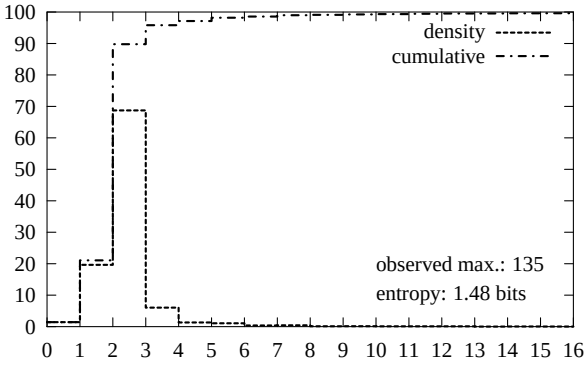

Figure 1: Empirical probability of a caller phrase starting *x* words into the message

These observations strongly suggest that when extracting caller phrases, positional cues should be taken into account. This is good news, especially since intrinsic features of the caller phrase may not be as reliable: a caller phrase is likely to contain names that are problematic for an automatic speech recognizer. While this is less of a problem when evaluating on manual transcriptions, the experience reported in (Huang et al., 2001) suggests that the relatively high error rate of speech recognizers may negatively affect performance of caller name extraction on automatically generated transcripts. We therefore avoid using anything but a small number of greetings and commonly occurring words like 'hi', 'this', 'is' etc. and a small number of common first names for extracting caller phrases and use positional information in addition to word-based features.

We locate caller phrases by first identifying their start position in the message and then predicting the length of the phrase. The empirical distribution of caller phrase lengths in the development data is shown in Figure 2. Most caller phrases are between two and four words long ('it's Pat', 'this is Pat Caller') and there are moderately good lexical indicators that signal the end of a caller phrase ('I', 'could', 'please', etc.). Again, we avoid the use of names as features and rely on a small set of features based on common words, in addition to phrase length, for predicting the length of the caller phrase.

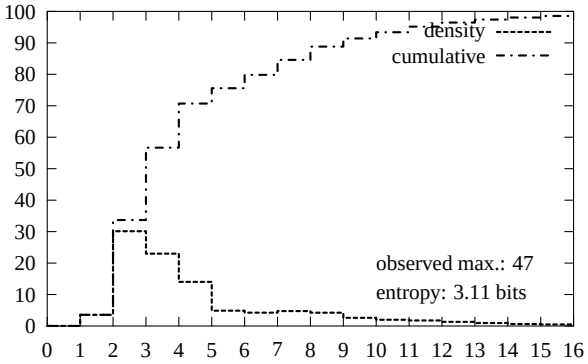

Figure 2: Empirical probability of a caller phrase being *x* words long

We have thus identified two classes of features that allow us to predict the start of the caller phrase relative to the beginning of the message, as well as the end of the caller phrase relative to its start. Since we are dealing with discrete word indices in both cases, we treat this as a classification task, rather than a regression task. A large number of classifier learners can be used to automatically infer classifiers for the two subtasks at hand. We chose a decision tree learner for convenience and note that this choice does not affect the overall results nearly as much as modifying our feature inventory.

Since a direct comparison to the log-linear named entity tagger described in (Huang et al., 2001) (we refer to this approach as *HZP log-linear* below) is not possible due to the use of different corpora and annotation standards, we applied a similar named entity tagger based on a log-linear model with trigram features to our data (we refer to this approach as *Col log-linear* as the tagger was provided by Michael Collins). Table 1 summarizes precision (P), recall (R), and F-measure (F) for three approaches evaluated on manual transcriptions: row *HZP log-linear* repeats the results of the best model from (Huang et al., 2001); row *Col log-linear* contains the results we obtained using a similar named entity tagger on our own data; and row *JA classifiers* shows the performance of the classifier method proposed in this section.

Like Huang et al. (2001), we count a proposed caller phrase as correct if and only if it matches the annotation of the evaluation data perfectly. The

numbers could be made to look better by using containment as the evaluation criterion, i.e., we would count a proposed phrase as correct if it contained an actual phrase plus perhaps some additional material. While this may be more useful in practice (see below), it is not the objective that was maximized during training, and so we prefer the stricter criterion for evaluation on previously annotated transcripts.

	P	R	F
HZP log-linear	.89	.80	.84
Col log-linear	.83	.78	.81
JA classifiers	.73	.68	.71

Table 1: Performance of caller phrase extraction (manual transcriptions)

While the results for the approach proposed here appear clearly worse than those reported by Huang et al. (2001), we hasten to point out that this is most likely not due to any difference in the corpora that were used. This is corroborated by the fact that we were able to obtain performance much closer to that of the best, finely tuned log-linear model from (Huang et al., 2001) by using a generic named entity tagger that was not adapted in any way to the particular task at hand. The log-linear taggers employ *n*-gram features based on family names and other particular aspects of the development data that do not necessarily generalize to other settings, where the family names of the callers may be different or may not be transcribed properly. In fact, it seems rather likely that the log-linear models and the features they employ over-fit the training data.

This becomes clearer when one evaluates on unseen transcripts produced by an automatic speech recognizer (ASR),[3] as summarized in Table 2. Rows *HZP strict* and *HZP containment* repeat the figures for the best model from (Huang et al., 2001) when evaluated on automatic transcriptions. The difference is that *HZP strict* uses the strict evaluation criterion described above, whereas *HZP containment* uses the weaker criterion of containment, i.e., an extracted phrase counts as correct if it contains exactly one whole actual phrase. Row *JA containment* summarizes the performance of our approach when evaluated on 101 unseen automatically transcribed messages. Since we did not have any labeled automatic transcriptions available to compare with the predicted caller phrase labels using the strict criterion, we only report results based on the weaker criterion of containment. In fact, we count caller phrases as correct as long as they contain the full name of the caller, since this is the common denominator in the otherwise somewhat heterogeneous labeling of our training corpus; more on this issue in the next section.

	P	R	F
HZP strict	.24	.16	.19
HZP containment	.73	.41	.52
JA containment	.74	.66	.70

Table 2: Performance of caller phrase extraction (automatic transcriptions)

The difference between the approach in (Huang et al., 2001) and ours may be partly due to the performance of the ASR components: Huang et al. (2001) report a word error rate of 'about 35%', whereas we used a recognizer (Bacchiani, 2001) with a word error rate of only 23%. Still, the reduced performance of the HZP model on ASR transcripts compared with manual transcripts is points toward overfitting, or reliance on features that do not generalize to ASR transcripts. Our main approach, on the other hand, uses classifiers that are extremely knowledge-poor in comparison with the many features of the log-linear models for the various named entity taggers, employing no more than a few dozen categorical features.

3.2 Caller Names

Extracting an entire caller phrase like 'this is Pat Caller' may not be all that relevant in practice: the prefix 'this is' does not provide much useful information, so simply extracting the name of the caller should suffice. This is more or less a problem with the annotation standard used for marking up voicemail transcripts. We decided to test the effects of changing that standard post hoc. This was relatively easy to do, since proper names are capitalized in the message transcripts. We heuristically identify caller names as the leftmost longest contiguous sub-

[3] An automatic transcription is the single best word hypothesis of the ASR for a given voicemail message.

sequence of capitalized words inside a marked-up caller phrase. This leaves us with 6731 messages with caller names in our development data.[4]

As we did for caller phrases, we briefly examine the distributions of the start position of caller names (see Figure 3) as well as their lengths (see Figure 4). Comparing the entropies of the empirical distributions with the corresponding ones for caller phrases suggests that we might be dealing with a simpler extraction task here. The entropy of the empirical name length distribution is not much more than one bit, since predicting the length of a caller name is mostly a question of deciding whether a first name or full name was mentioned.

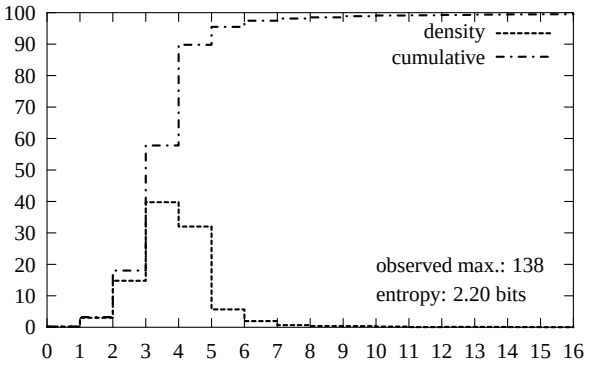

Figure 3: Empirical probability of a caller name starting *x* words into the message

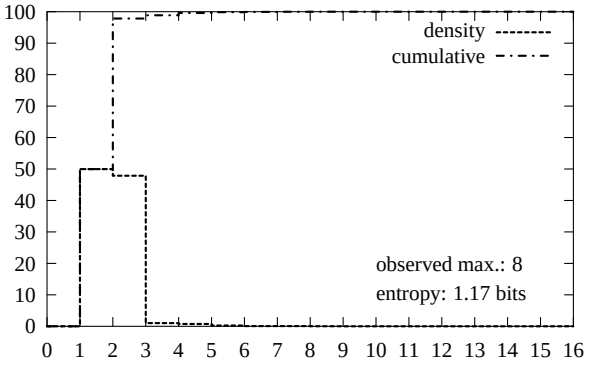

Figure 4: Empirical probability of a caller name being *x* words long

[4]The vast majority of messages that do not mention a name as part of their caller phrase employ the caller phrase 'it's me', which would be easy to detect and treat separately.

The performance comparison in Table 3 shows that we are in fact dealing with a simpler task. Notice however that our method has not changed at all. We still use one classifier to predict the beginning of the caller name and a second classifier to predict its length, with the same small set of lexical features that do not include any names other than a handful of common first names.

	P	R	F
phrase	.730	.684	.706
name	.860	.871	.865

Table 3: Caller phrase vs. name extraction (manual transcriptions)

4 Phone Numbers

The development data contain 5303 marked-up phone numbers, for an average of almost 0.7 phone numbers per non-empty message. These phone numbers fall into the following categories based on their realization:

- 4472 (84%) consist exclusively of spoken numbers
- 679 (13%) consist of spoken numbers and the words 'area', 'code', and 'extension'
- 152 (3%) have additional material, due to corrections, hesitations, fragments, and questionable markup

Note that phone numbers in the North American Numbering Plan are either ten or seven digits long, depending on whether the Numbering Plan Area code is included or not. Two other frequent lengths for phone numbers in the development data are four (for internal lines) and, to a lesser extent, eleven (when the long distance dialing prefix is included, as in 'one eight hundred . . . ').

This allows us to formulate the following baseline approach: find all maximal substrings consisting of spoken digits ('zero' through 'nine') and keep those of length four, seven, and ten. Simple as it may seem, this approach (which we call *digits* below) performs surprisingly well. Its precision is more than 78%, partly because in our corpus there do not

occur many seven or ten digit numbers that are not phone numbers.

Named entity taggers based on conditional models with trigram features are not particularly suited for this task. The reason is that trigrams do not provide enough history to allow the tagger to judge the length of a proposed phone number: it inserts beginning and end tags without being able to tell how far apart they are. Data sparseness is another problem, since we are dealing with 1000 distinct trigrams over digits alone, so a different event model that replaces all spoken digits with the same representative token might be better suited, also because it avoids overfitting issues like accidentally learning area codes and other number patterns that are frequent in the development data.

However, there is a more serious problem. Even if the distance between the start and end tags that a named entity tagger predicts could be taken into account, this would not help with all spoken renditions of phone numbers. For example, '327-1025' could be read aloud using only six words ('three two seven ten twenty five'), and might be incorrectly rejected because it appears to be of a length that is not very common for phone numbers.

We therefore approach the phone number extraction task differently, using a two-phase procedure. In the first phase we use a hand-crafted grammar to propose candidate phone numbers. This avoids all of the problems mentioned so far, yet the complexity of the task remains manageable because of the rather simple structure of most phone numbers in our development data noted above. The advantage is that it allows us to simultaneously convert spoken digits and numbers to a numeric representation, whose length can then be used as an important feature for deciding whether to keep or throw away a candidate. Note that such a conversion process is desirable in any case, since a text-based application would presumably want to present digit strings like '327-1025' to a user, rather than 'three two seven ten twenty five'. This conversion step is not entirely trivial, though: for example, one might transcribe the spoken words 'three hundred fourteen ninety nine' as either '300-1499' or '314.99' depending on whether they are preceded by 'call me back at' vs. 'I can sell it to you for', for example. But since we are only interested in finding phone numbers, the extraction component can treat all candidates it proposes as if they were phone numbers.

Adjustments of the hand-crafted grammar were only made in order to increase recall on the development data. The grammar should locate as many actual phone numbers in the development corpus as possible, but was free to also propose spurious candidates that did not correspond to marked-up phone numbers. While it has recently been argued that such separate optimization of recall and precision is generally desirable for certain learning tasks (Agarwal and Joshi, 2001; Joshi et al., 2001), the main advantage in connection with hand-crafted components is simplified development. Since we noted above that 97% of all phone numbers in our development data are expressed fairly straightforwardly in terms of digits, numbers, and a few other words particular to the phone number domain, we might expect to achieve recall figures close to 97% without doing anything special to deal with the remaining 3% of difficult cases. It was very easy to achieve this recall figure on the development data, while the ratio of proposed phone numbers to actual phone numbers was about 3.2 at worst.[5]

A second phase is now charged with the task of weeding through the set of candidates proposed during the first phase, retaining those that correspond to actual phone numbers. This is a simple binary classification task, and again many different techniques can be applied. As a baseline we use a classifier that accepts any candidate of length four or more (now measured in terms of numeric digits, rather than spoken words), and rejects candidates of length three and less. Without this simple step (which we refer to as *prune* below), the precision of our hand-crafted extraction grammar is only around 30%, but by pruning away candidate phone numbers shorter than four digits precision almost doubles while recall is unaffected.

We again used a decision tree learner to automatically infer a classifier for the second phase. The features we made available to the learner were the length of the phone number in numeric digits, its

[5] It would of course be trivial to achieve 100% recall by extracting all possible substrings of a transcript. The fact that our grammar extracts only about three times as many phrases as needed is evidence that it falls within the reasonable subset of possible extraction procedures.

distance from the end of the message, and a small number of lexical cues in the surrounding context of a candidate number ('call', 'number', etc.). This approach (which we call *classify* below) increases the precision of the combined two steps to acceptable levels without hurting recall too much.

A comparison of performance results is presented in Table 4. Rows *HZP rules* and *HZP log-linear* refer to the rule-based baseline and the best log-linear model of (Huang et al., 2001) and the figures are simply taken from that paper; row *Col log-linear* refers to the same named entity tagger we used in the previous section and is included for comparison with the HZP models; row *JA digits* refers to the simple baseline where we extract strings of spoken digits of plausible lengths. Our main results appear in the remaining rows. The performance of our hand-crafted extraction grammar (in row *JA extract*) was about what we had seen on the development data before, with recall being as high as one could reasonably expect. As mentioned above, using a simple pruning step in the second phase (see *JA extract + prune*) results in a doubling of precision and leaves recall essentially unaffected (a single fragmentary phone number was wrongly excluded). Finally, if we use a decision tree classifier in the second phase, we can achieve extremely high precision with a minimal impact on recall. Our two-phase procedure outperforms all other methods we considered.

	P	R	F
HZP rules	.81	.83	.82
HZP log-linear	.90	.83	.86
Col log-linear	.88	.93	.91
JA digits	.78	.70	.74
JA extract	.30	.96	.45
JA extract + prune	.59	.96	.73
JA extract + classify	.94	.94	.94

Table 4: Performance of phone number extraction (manual transcriptions)

We evaluated the performance of our best models on the same 101 unseen ASR transcripts used above in the evaluation of the caller phrase extraction. The results are summarized in Table 5, which also repeats the best results from (Huang et al., 2001), using the same terminology as earlier: rows *HZP strict* and *HZP containment* refer to the best model from (Huang et al., 2001) – corresponding to row *HZP log-linear* in Table 4 – when evaluated using the strict criterion and containment, respectively; and row *JA containment* refers to our own best model – corresponding to row *JA extract + classify* in Table 4.

	P	R	F
HZP strict	.56	.52	.54
HZP containment	.85	.79	.82
JA containment	.95	.94	.95

Table 5: Performance of phone number extraction (automatic transcriptions)

It is not very plausible that the differences between the approaches in Table 5 would be due to a difference in the performance of the ASR components that generated the message transcripts. From inspecting our own data it is clear that ASR mistakes inside phone numbers are virtually absent, and we would expect the same to hold even of an automatic recognizer with an overall much higher word error rate. Also, for most phone numbers the labeling is uncontroversial, so we expect the corpora used by Huang et al. (2001) and ourselves to be extremely similar in terms of mark-up of phone numbers. So the observed performance difference is most likely due to the difference in extraction methods.

5 Conclusion and Outlook

The novel contributions of this paper can be summarized as follows:

- We demonstrated empirically that positional cues can be an important source of information for locating caller names and phrases.
- We showed that good performance on the task of extracting caller information can be achieved using a very small inventory of lexical and positional features.
- We argued that for extracting telephone numbers it is extremely useful to take the length of their numeric representation into account. Our grammar-based extractor translates spoken numbers into such a numeric representation.

- Our two-phase approach allows us to efficiently develop a simple extraction grammar for which the only requirement is high recall. This places less of a burden on the grammar developers than having to write an accurate set of rules like the baseline of (Huang et al., 2001).
- The combined performance of our simple extraction grammar and the second-phase classifier exceeded the performance of all other methods, including the current state of the art (Huang et al., 2001).

Our results point towards approaches that use a small inventory of features that have been tailored to specific tasks. Generic methods like the named entity tagger used by Huang et al. (2001) may not be the best tools for particular tasks; in fact, we do not expect the bigram and trigram features used by such taggers to be sufficient for accurately extracting phone numbers. We also believe that using all available lexical information for extracting caller information can easily lead to over-fitting, which can partly be avoid by not relying on names being transcribed correctly by an ASR component.

In practice, determining the identity of a caller might have to take many diverse sources of information into account. The self-identification of a caller and the phone numbers mentioned in the same message are not uncorrelated, since there is usually only a small number of ways to reach any particular caller. In an application we might therefore try to use a combination of speaker identification (Rosenberg et al., 2001), caller name extraction, and recognized phone numbers to establish the identity of the caller. An investigation of how to combine these sources of information is left for future research.

Acknowledgements

We would like to thank Michiel Bacchiani, Michael Collins, Julia Hirschberg, and the SCANMail group at AT&T Labs. Special thanks to Michiel Bacchiani for help with ASR transcripts and to Michael Collins for letting us use his named entity tagger.

References

Ramesh C. Agarwal and Mahesh V. Joshi. 2001. PNrule: A new classification framework in data mining (A case study in network intrusion detection). In *First SIAM International Conference on Data Mining*, Chicago, IL.

Michiel Bacchiani. 2001. Automatic transcription of voicemail at AT&T. In *International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT.

Abhay Bhushan, Ken Pogran, Ray Tomlinson, and Jim White. 1973. Standardizing network mail headers. Internet RFC 561.

David H. Crocker. 1982. Standard for the format of ARPA internet text messages. Internet RFC 822, STD 11.

Yoshihiko Gotoh and Steve Renals. 2000. Information extraction from broadcast news. *Philosophical Transactions of the Royal Society of London, Series A*, 358:1295–1310.

Julia Hirschberg, Michiel Bacchiani, Don Hindle, Phil Isenhour, Aaron Rosenberg, Litza Stark, Larry Stead, Steve Whittaker, and Gary Zamchick. 2001. SCANMail: Browsing and searching speech data by content. In *7th European Conference on Speech Communication and Technology*, Aalborg, Denmark.

Jing Huang, Geoffrey Zweig, and Mukund Padmanabhan. 2001. Information extraction from voicemail. In *39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. 2001. Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *ACM SIGMOD International Conference on Management of Data*, Santa Barbara, CA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Empirical Methods in Natural Language Processing*, Philadelphia, PA.

Aaron Rosenberg, Julia Hirschberg, Michiel Bacchiani, S. Parthasarathy, Philip Isenhour, and Larry Stead. 2001. Caller identification for the SCANMail voicemail browser. In *7th European Conference on Speech Communication and Technology*, Aalborg, Denmark.