Automatic Content Extraction for Voicemail

Steven E. Czerwinski and Barbara Hohlt
CS294-1 *Multimedia Systems and Applications*
Fall 1999
Computer Science Division
University of California, Berkeley
{czerwin,hohltb}@cs.berkeley.edu

1 Introduction

The Age of Information is upon us. Every day, more and more devices are being hooked up to the global information network, which partly consists of the telephony, Internet, and pager networks. Every day, more and more content providers are pushing information out to users. More so than ever, people are expected to keep up with this information, and act on it accordingly. Because of this, timely and efficient access to a user's information has become vital.

In this project, we investigated one particular form of information the user receives: voicemail. The traditional way of receiving and viewing voicemails through a phone is clumsy and does not lend itself to high speed browsing – we thought something better could be done. To this end, we have designed and built the *Media Manager* with the goal of increasing the user's ability to access and view his voicemail from different types of devices, including purely audio devices, display screens, and a combination of the two. We have done this by investigating and building different types of audio processing tools to automatically extract important content from voicemail audio files.

There are two main sources of information that can be retreived from a voice message: (1) the natural language content and (2) the pitch pause, intensity, and intonation information known as *prosody* [5]. The first type of information can be retreived through speech recognition and can be used to recognize key words or phrases, along with displaying the voicemail in a text. With the prosody information, it is possible to speed up a message by removing pauses and detect important events with pitch emphasis.

In this paper, we will discuss the design and implementation of the *Media Manager* and its audio processing algorithms for extracting content from voicemail. The *Media Manager* is implemented as a general purpose email system (voicemail is MIME encoded email) with special enhancements for dealing with audio files containing speech. The audio processing algoritms we use in the project include pause detection for increasing playback speeds of voicemails, pitch emphasis detection for recognizing important words, speech recoginition for translation, and natural language processing for extracting key words. The *MediaManager* was developed on top of the Ninja [4] infrastructure.

2 Related Work

Creating tools to allow for more efficient skimming and browsing of speech files has been researched by several different groups. One of the earlier and most comprehensive projects was the SpeechSkimmer [2] done at MIT's Multimedia Lab. This project created general infrastructure and navigation tools to efficiently browse collections of large speech files. It employed several different audio processing techniques to recover the structure of the speech file, along with creating different representations of the file for faster playback and navigation. These techniques included pause removal for faster playback, pitch emphasis detection for highlighting important sections, and phase vocoding for improved playback of time compressed audio.

Much of the low-level audio processing tools used in this project are based on the work done on the SpeechSkimmer, since the aims of the browsing features were so similar. However, by the nature of the processing algorithms, the our system did not see as good results because of the small size of the voicemails.

Another related project [5] was done at Microsoft Research, in which they explored the usefulness of autosummarizing multimedia presentations. These presentations included slides and the audio and video of the original presentation. The goal of the project was to use several different types of hints (including slide translations, pitch emphasis, and user access patterns) to generate summaries for the presentations. They found that they could produce summaries 25% the size of the original presentation that would give users a "fair" amount of information.

The Iceberg [13] project at UC Berkeley is exploring a common infrastructure to support both voice and data services. Instead of treating telephony and Internet services as separate entities, this project would unify the two into one, more useful system. They are currently exploring such services as the Universal Inbox [11] which would present users their voicemail and email through one interface. They are working on simple strategies for translating voicemails to text for better textual presentation, but they are not actively exploring different representations as this project is.

Finally, the increased profileration of audio content due to the Internet has motivated several new projects aimed at extracting information from audio files. These projects range in size and shape, but all are trying to improve audio content processing tools, just as our project does. For example, the ClueVideo System [12] at IBM has created tools to differentiate audio and music, to aide in segmenting presentations.

3 Audio and Speech Tools

In this section, we will discuss the various methods we used to extract information content from the audio files representing the voicemails we wish to browse. Throughout the course of this project, we have explored processing the audio files on many different levels, from the lowest level of examining the sound energy in each frame of audio, to the highest level of using complex natural language processing techniques. The individual methods we are presenting are not novel, but using them all together in the context of voicemail is an area of current research.

3.1 Pause detection

Pause detection is one of the simplest and easiest ways to collect voice information from audio. The goal of the pause detection is to find and measure the boundaries between the audio representing individual words. Explained in a different way, pause detection tries to classify each audio frame into one of two categories: speech or silence. One might think this should be an easy task, accomplished by just looking at the magnitude of the audio energy in each frame. However, because of background noise, inaccuracies in recording, pause detection is not as simple as thresholding the magnitude.

Why do we need pause detection? Even though it represents a very simple form of information, it does aide in several aspects of this project. The first major contribution it makes is it can be used to reduce playback times of voicemail audio, which is a key feature for navigating a large set of voicemails. This reduction is accomplished by simply removing all the pauses in a given piece of audio. As shown by [3], by removing the pauses, we can typically reduce the playback time by 20-30%, while maintaining intelligibility of the speech. Listening to this form of skimmed audio can be perceived to be exhausting if the segment is too long [3], but for the purposes of voicemail, it is adequate.

Secondly, pause detection can be used to detect talk spurts. For the purpose of this paper, we define talk spurts to be a series of spoken words that should be grouped together because of semantic meaning. Essentially, talk spurts can be thought of as sentences.

By examining the length of pauses between words, we can recover information about the talk spurts, and therefore gain information about the sentence structure of the voicemail. This gives us a very useful organizational tool, allowing us to better present the transcript of the voicemail to the user. We can also segment the playback, so that the user can playback a single talk spurt at a time, rather than playing back the complete voicemail.

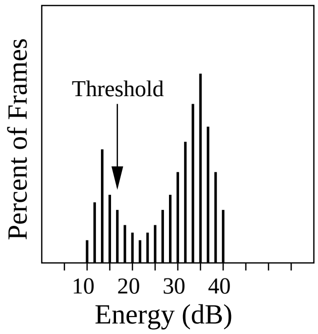

Figure 1: Choosing the Pause threshold

3.1.1 Pause detection algorithm

In order to detect pauses in the the voicemail audio files, we implemented an algorithm based on the work done by [2, 8]. Simply explained, the algorithm detects speech based on thresholding the audio energy in each frame, where the threshold is adaptively chosen to handle noise, and then using heuristics based on language information to remove artifacts.

In more detail, the algorithm first chooses a threshold value by examining the smoothed histogram of the audio energy magnitude in each 10 ms frame. The histogram is smoothed by taking a three point average for each bin. Figure 1 shows one sample histogram. The threshold is chosen by adding 5 dB to the first peak in the histogram. There are usually two peaks in the histogram, representing the sound levels of background noise (the first peak) and the speech.

Then, each 10 ms audio frame is examined, marking it as speech if its energy level is greater than the threshold. We also mark a frame as speech if it has a high number of zero-crossings (ZCR) [10] (greater than a fixed value), since this is usually indictative of speech. After this, we make several passes, modifying the classification of the frames as determined by a set of heuristics [2]. The heuristics are created based on speech models. For example, one heuristic says to remove any 20 ms or less gap in a speech region, since it is too short to be a real pause. Similarly, a region of speech that is less than 50 ms is removed since it is far too short be a real word.

We will discuss the speed and accuracy of this algorithm in Section 5.1.

3.2 Pitch detection

Pitch is another low level audio processing technique we used to extract information from the voicemail audio files. The goal of pitch detection is to determine the pitch of each unit of speech in the audio file. We use this information to find important words in the voicemail, because, in natural speech, rises in pitch usually signify emphasized words and topic introduction [6].

Accurate pitch detection is difficult for the same reasons pause detection is difficult: background noise and errors introduced by recording. Additionally, the average pitch of spoken words varies from person to person,

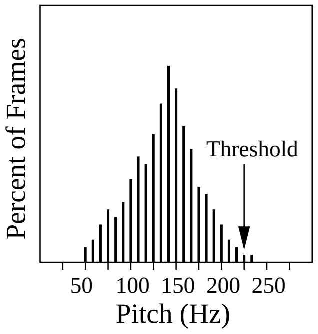

Figure 2: Choosing the Pitch threshold

especially across gender. The pitch detection algorithm must be robust to both noise and changes in speakers.

In order to detect rises in pitch, we implemented an algorithm similar to [2]. This algorithm finds rises in pitch for one second intervals, and is robust against changes in speakers.

The algorithm works as follows: First, the audio is divided into 20 ms frames, and the pitch is calculated for each frame. The algorithm originally presented in [2] called for 10 ms frames, but as other works suggested 20 ms intervals lead to better performance because pitch periods are typically 15 ms. The pitch for a given frame is determined by doing a 1024 point Discrete Fourier Transform (DFT) [9], followed by an application of Subharmonic Summation (SHS). The SHS is used to make sure the lower harmonics receive contributions from their higher frequency counterparts.

After all the pitch values are determined, a histogram is created with a bin size of 1 Hz. Figure 2 shows a typical pitch histogram. Using the histogram, a pitch threshold is chosen such that only 1% of the frames have a pitch higher than it. By doing this, we have chosen a pitch threshold that represents unusually high pitches for the speaker, assuming there is only one speaker in the given audio. Then each 1 second interval is examined, and if an interval has more than 3 frames with pitch higher than the interval, it is marked as emphasized.

The performance and problems with this algorithm will be discussed in Section 5.2.

3.3 Speech detection

Speech detection is a higher level audio processing tool that attempts to translate the audio of spoken words into their text counterparts. It is typically accomplished using some form of mathematical model of speech (modeling individual utterances or syllables) along with classification algorithms such as hidden Markov Models or neural networks. Additionally, most speech systems incorporation some sort of grammar and word dictionary to predict the probability of word combinations.

In our project, we decided to experiment with the use-fullness of using state of the art speech detection tools to translate voicemails to text. If this could be done perfectly, it would negate the need for listening to voicemail at all. Our experiences and results of using speech detection will be discussed in 5.3.

3.4 Natural Language Processing

Natural Language Processing (NLP) [1] is area of Artificial Intelligence research with the goal of enabling computers to "understand" natural language (human dialogue). Typically, these systems take in dialogues in the form of text, and then parse them to gain some information about the content of the dialogue. The parsing and comprehension is done using a combination of grammars describing sentence structure (both syntax and semantics) along with some decision making tools to judge best fits. NLP is still a very active area of research, with the promise of some day allowing humans to interact with computers through the more comfortable interface of natural language.

For our project, we used NLP techniques to enable the computer to understand some very common sentences spoken in voicemail messages. For example, we created grammars that would parse and tag sentences that conveyed phone numbers or the identity of the caller. In some sense, it is a form of word spotting, but it more sophisticated since it relies on knowledge of sentence structure and grammar rules.

4 Architecture

4.1 Media Manager Service

The Media Manager service is implemented as a Ninja iSpace [14] service. Its purpose is to handle email, both regular email and voicemail encoded in MIME, along with our own extensions for transcoding the voicemail. We have implemented each result type of transcoder as a new MIME type. The Media Manager works across any mail protocol. As Figure 3 shows, the `MediaManager` implements the Media Manager interface. Through this interface it receives requests from clients and in turn accesses the actual mail message via the Mail Access interface. The Mail Access interface is a generic interface used to generalized mail access through different mail protocols. In this setup , users can access all their mail across different mail protocols via the Media Manager. The `MediaManager` will also use the `Transcoder` service to perform needed transformations on mail messages parts.

The Media Manager interface exports methods which allow clients to retrieve their mail folders, list the contents, retrieve messages, and transcode the contents of those messages. We of course, are interested in transcoding the contents of voicemail.

The Mail Access interface may be implemented for any mail protocol, such as NinjaMail [14], POP, and IMAP. Currently only NinjaMail Access is implemented. The `FolderStore` contains records of every protocol where a user has mail folders.

`MediaMessages`, the type of messages used in our system, are simplified versions of regular mail messages and are meant to be used with simpler clients. Only a few headers are handled and only one level of MIME parts is supported. The parts of a message are referred to as `ContentObjects` and are specified by their `ContentIDs`. Each `ContentObject` has a MIME type.

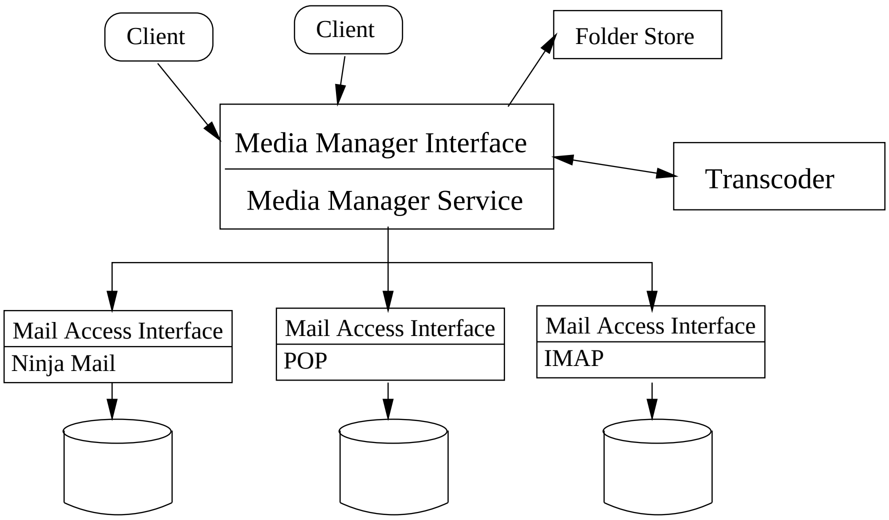

Figure 3: System Architecture

The `MediaManager` supports text, audio, and multipart MIME types. In addition we have our own MIME extensions; skimmed audio, text summary, transcript, outline, and audio summary. By passing a `ContentID` and the desired MIME type to the `MediaManager`, a client may retrieve a `ContentObject` returned as a particular MIME type. The `Transcoder` service does the actual transcoding. This is useful for clients who do not have speakers, for example, and need their voicemail displayed as text.

4.2 Transcoder Service

The `Transcoder` service is a Ninja `iSpace` service which transforms data. It is responsible for transforming data into plain text, transcript, text summary, text outline, plain audio, audio summary, and skimmed audio. Transcript represents content that has been generated by doing speech recognition. Text summary represents automatic text summarization done by extracting the emphasized words found by the techniques described in Section 3.2 and key words found using NLP. Outline organizes a text summary into an outlined form, which is generated by using the talk spurts of the audio file. Plain audio may represent original audio or that which has been synthesized from a text email. Audio summary is audio generated from the text summary. Skimmed audio is audio that has had all pauses removed.

The basic transformations from speech to text and text to speech use programs developed with IBM's speech recognizer and synthesizer software development kit [7]. We used the Speech Recognition Control Language (S-RCL) to define a grammar which recognizes names and phone numbers for the text summaries. The SRCL also has an annotation feature which allows key words and phrases to be tagged when the grammar is defined. Both the annotation and recognized phrase are returned to the application. This simplifies the task of parsing of the results for meaningful information. In order to set our input and output sources to and from the speech engine, we also created a custom audio library.

Component	*Processing time*	*Accuracy*
Pause detection	100 ms	70%
Talk spurts	100 ms	80%
Pitch detection	2 secs	40%
Speech recognition	19 secs	40%
NLP	-	80%

Table 1: Results of components

5 Results

Unfortunately, we were not able to test and benchmark the performance and accuracy of the `MediaManager` as thoroughly as we would have wished. In the next several subsections, we have some preliminary numbers and estimates of the accuracy of the individual components.

Table 1 shows a listing of the execution times and accuracy levels of the major components of the system. The accuracy levels were not

5.1 Pause Detection

As Table 1 shows, the execution time for the pause detection algorithm is pretty fast, mainly because it is a computationally trivial algorithm. In fact, pause detection algorithms were initially developed 15-20 years ago to detect beginnings and endings of recordings. Available computation power on user devices has greatly increased since then.

The accuracy of the pause detection algorithm is fairly good, typically 70%. Usually, the pause detector failed on words with syllables that have lower energy or pauses in them, such as *s* or *ou* sounds. We should point out that, because of the components using pause detection, even a performance level of 70% leads to the perception of very good performance. This is because for algorithms like pause removal, you don't have to remove every single pause. In terms of playback speedup, we found that we reduced the playing time of a voicemail to be between 50 and 70% of the original.

We also saw very good performance from the talk spurt detector, only failing to correctly mark a spurt 20% of the time when dealing with naturally spoken speech. Usually, the talk spurt detector had problems dealing with long pauses introduced by the speaker as he or she thinks (i.e., a lot of *umms* and *ahhs*).

5.2 Pitch Detection

The pitch emphasis detector did not perform as well as expected. As Table 1 indicates, we saw about 40% accuracy of the words detected as emphasized. To further break this down, most of that inaccuracy was due to false positives, i.e., words that were marked as being emphasized when they were, in fact, not important in the message. These type of errors accounted for 80% of the classification errors (the other type of error was when the detector missed marking what we deemed as an important word in the voicemail).

Many of the false positives were either due to emphasis created by new topic introduction, or because of high pitch sounds in the underlying syllables. As [6] points out, increases in pitch can signify either topic introduction or emphasized words. Unfortunately, we would like to only capture the emphasized words, not the topic introduction. Words like *also*, *and*, and *oh* were often highlighted because of this.

Underlying high pitch sounds also threw off the emphasis detector, because they appeared to the thresholding mechanism as increases in the average pitch. For example, words with a lot high frequency sounds like *services* and *sepulveda* often were categorized as emphasized due to their *s* sounds. The word *sepulveda* has especially bad because it also threw off the pause detector, since the long *v* and *d* sounds have low energy for extend periods.

5.3 Speech Recognition

The speech recognition software also did not perform as well as we had hoped. We saw accuracy rates as high as 90% when performing dictations, but when recording voicemails in conversation tones, the accuracy dropped significantly to about 30 to 40%.

5.4 Natural Language Procesing

The NLP components of our system performed fairly well, mainly because the speech recognition software was really able to zero in on some expected forms of sentences. If we included a sentence in the voice mail that fit a grammar rule in the NLP, it was correctly identified 80% of the time.

6 Conclusion

In this paper, we have described the design and implementation of the *Media Manager*, a voicemail-aware email system. Our work on this project has focused on developing algorithms and transcoders to extract useful content from voicemail audio files, in order to enhance the voicemail presentation to the user through both audio and graphic display devices. To this end, we have investigated the use of audio processing techniques including pause detection to gain sentence structure and to speedup playback, pitch detection to highlight important parts of the voicemail, speech recognizer to create transcripts of the voicemails, and natural language processing (NLP) tools to extract high level information from the voicemail.

In general, we have learned much about the strengths and weaknesses of each of these tools. We have found that pause detection works fairly well for recognizing sentence structure and for reducing playback time, but pitch detection does not work well because of false positives introduced by transitions and high pitch sounds. Similarly, the state of the art speech recognizers do perform well in dictation settings, but their performance needs to be improved in conversation settings. Finally, NLP techniques hold a lot of promise for handling common types of voicemail, but constructing the grammars and dictionaries for them is a daunting task.

7 Future Directions

There is still much room for future work in this area. For instance, we would have liked to have investigated the use of speech recognition and NLP feedback in the detecting emphasis. The pitch detector did not perform as well as it could have because of false positives introduced due to topic transitions and high-pitched syllables. If we had used information about the word content in the regions identified by the pitch detector emphasis, we may have been able to reduce these false positives. (i.e., if the NLP identifies the area as a topic introduction, then it should not be emphasized because these introduction words such as *and* and *oh* are not interesting).

Another area of future work is to improve the available speech recognition software. IBM's ViaVoice is a very useful tool, but it could provide developers much more detailed information, such as word recognition confidence levels, and the timings for the individual words. Using this information, we could have built UIs that highlighted poorly-recognized words for the user, and allowed them to click on the individual word to hear its audio. The start and end timings for the individual words would have also been useful in picking off the emphasized words.

Finally, we believe this project could have greatly benefitted from a better NLP grammar developed specifically for voicemails. Voicemails do represent a very specialized form of speech communication, and by focusing the speech recognizer on the expected sentence forms, we would have improved recognition for both the transcripts and NLP features.

We would like to conclude by pointing out that even though the audio tools are not perfect, we found that information extracted by the *Media Manager* did aide in our ability to understand and navigate through the voicemails. With the increase of communication in the form of email, cell phones, and pagers, having tools such as this one will greatly aide users in the future.

References

[1] Allen, J. *Natural Language Understand.* Benjamin Cummings Publishing Company, 1987.

[2] Arons, B. *Interactively Skimming Recored Speech.* PhD thesis, MIT, 1985.

[3] Foulke, E. *Perception of Language.* Charles E. Merill Publishing Company, 1971, ch. The Perception of Time Compressed Speech, pp. 79–107.

[4] Gribble, S. D., Welsh, M., Brewer, E. A., and Culler, D. The multispace: an evlolutionary platform for infrastructural services. In *Proceedings of the 1999 Usenix Annual Technical Conference* (Monterey, CA, June 1999), Usenix.

[5] He, L., Sanocki, E., Gupta, A., and Grudin, J. Auto-summarization of audio-video presentations. In *Proceedings of the 1999 ACM Multimedia Conference* (Orlando, FL, October 1999), ACM.

[6] Hirschberg, H., and Grosz, B. Intonational Features of Local and Global Discourse. In *Proceedings of the Speach and Natural Lanuage workshop* (Harriman, NY, Feb 1992), Morgan Kaufman Publishers.

[7] IBM. ViaVoice SDK. `http://www-4.ibm.com/software/speech/dev/sdk_linux.html`.

[8] Lamel, L. F., Rabiner, L. R., Rosenberg, A. E., and Wilson, J. G. An Improved Endpoint Detector for Isolated Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* (August 1981), 771–785.

[9] Oppenheim, A. V., and Schafer, R. W. *Discrete-Time Signal Processing.* Prentice Hall, 1989.

[10] O'Shaughnessy, D. *Speech Communications: Human and Machine.* Addison-Wesley Publishing Company, Inc., 1987.

[11] Raman, B. The Universal Inbox. `http://www.cs.berkeley.edu/~bhaskar/univ-inbox/`.

[12] Srinivasan, S., Petkovik, D., and Ponceleon, D. Towards robust features of clasifying audio in the cuevideo system. In *Proceedings of the 1999 ACM Multimedia Conference* (Orlando, FL, October 1999), ACM.

[13] The Iceberg Team. The Iceberg Project. `http://iceberg.cs.berkeley.edu`.

[14] The Ninja Team. The Ninja Project. `http://ninja.cs.berkeley.edu`.