Proceedings of the 8th Annual Conference of the International Speech Communication Association
(Interspeech 2007), pp.2617-2620, August 2007.

Automatic Transcription for a Web 2.0 Service to Search Podcasts

Jun Ogata, Masataka Goto, and Kouichirou Eto

National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, JAPAN

Abstract

This paper describes speech recognition techniques that enable a Web 2.0 service *"PodCastle"* where users can search and read transcribed texts of podcasts, and correct recognition errors in those texts. Most previous speech recognizers had difficulties transcribing podcasts because podcasts include various kinds of contents recorded in different conditions and cover recent topics that tend to have many out-of-vocabulary words. To overcome such difficulties, we continuously improve speech recognizers by using information aggregated on the basis of Web 2.0. For example, a language model is adapted to a topic of the target podcast on the fly, the pronunciations of out-of-vocabulary words are obtained from a Web 2.0 service, and an acoustic model is trained by using the results of the error correction by anonymous users. The experiments we report in this paper show that our techniques produce promising results for podcasts.
Index Terms: podcast, speech recognition, Web 2.0, acoustic model, language model

1. Introduction

One of the main applications of the automatic speech recognition of multimedia data is to serve as the basis for automatic indexation for an information retrieval system. From this viewpoint, spoken document retrieval (SDR) systems have been developed by many research groups across the world [1][2]. However, SDR systems or technologies have not been in practical use such as text retrieval engines (e.g., Google and Yahoo!). because the automatic transcription and indexation of multimedia data in the real world is very difficult. The amount of speech data is increasing rapidly on the web because *podcasts* that are often referred to as audio blogs have recently become popular and widespread. We therefore think SDR technologies and systems are in great demand especially for web applications or services.

As an application of SDR technologies based on speech recognition, we have developed a new web application, called *"PodCastle"*[3], in which podcasts can be searched, read, and annotated. PodCastle enables a user to search podcasts that include a search term and read automatically transcribed texts of the found podcast. Furthermore, it also enables the user to correct (annotate) recognition errors in the texts via an efficient error correction interface. If a lot of users contribute to the error correction, PodCastle can provide relatively accurate transcriptions of podcasts and improve the performance of full-text search. In addition, the speech recognizer can be trained from transcriptions whose recognition errors are corrected by users. Through this system architecture, we aim to improve not only the search performance but also the speech recognition performance.

In this paper, we present a podcast transcription system integrated into PodCastle. There are two main difficulties in transcribing podcasts. First, the contents and recording environments of podcasts have a wide variety. Second, podcasts tend to include words and phrases related to recent topics, which are usually not registered in the system vocabulary of state-of-the-art speech recognizers. To overcome these difficulties, we first introduce a method to keep the language model of our speech recognizer up-to-date by using on-line news texts. We then present three methods for improving the speech recognition performance by using information that is aggregated on Web 2.0 services: a language model adaptation method using RSS metadata of podcasts, a pronunciation acquisition method using an external Web 2.0 service that collects various keywords including recent ones, and a method of training the acoustic model of the recognizer from the results of user corrections to speech recognition errors.

The remainder of the paper is organized as follows. The next section provides an overview of PodCastle. Section 3 describes our podcast transcription system used as the baseline, and Section 4 presents the improving methods on the basis of Web 2.0. The last section provides conclusions and our future plans.

2. A new web application for searching, reading and annotating podcasts

Podcasting is a mechanism in which multimedia files distributed over the web can be downloaded automatically for playback on portable media players and personal computers, and it has become very popular recently. A podcast consists of several audio data (MP3 files) called *episodes* and a syndication feed (RSS) that includes metadata information about episodes. The feed provides not only the list of URLs of audio files by which episodes can be accessed but also other information such as the published date, titles, and summaries. With regard to podcasting, the amount of audio data on the web has been steadily increasing.

PodCastle aims to encourage the structuring of podcasts by using several technologies such as automatic transcription, full-text search, and efficient browsing. In this system, first, podcasts (i.e., audio data and RSS) are collected by a web crawler. Then, the audio data are transcribed by the speech recognizer to create indices for searching. The recognition results include not only a word sequence but also other information such as the beginning and ending time for each word, competitive candidates, and confidence scores. By using these recognition results and RSS feeds, PodCastle has three main functions as follows.

2.1. Searching function

This function performs the full-text searching of podcasts. The text indices used in searching are generated by the speech recognizer. When a user inputs a search term, the system displays a list of episodes that include the search term. Each episode is accompanied with a text excerpt around the highlighted search term and the user can listen to each excerpt. When the user clicks to select an episode, the system moves to the next *reading function*.

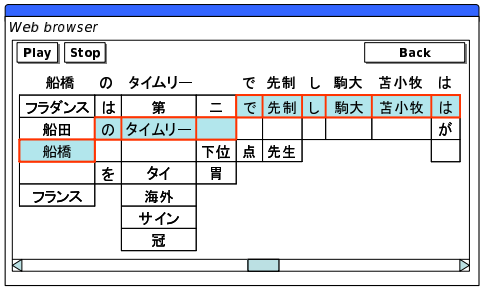

Figure 1: *The error correction interface in PodCastle.*

2.2. Reading function

This function enables a user to "read" a podcast by displaying the full-texts of a podcast episode. Therefore, the user can grasp the contents of the episode even without playback equipment for the audio data. In PodCastle, since all the transcribed texts are open to the public as *permalinks*, a user can search and find a podcast episode in the same manner as searching a web page on a general text-search engine. As a result, it leads to the recognition of a podcast by more and more users.

2.3. Annotating function

The unique feature of PodCastle is that all users accessing the PodCastle service can annotate the full text of a podcast. This is called the *annotating function*, and it refers to generating a full-text transcription of the podcast. However, transcribing manually from scratch is a very costly and time consuming process for users. Thus, this function provides an error correction interface in which recognition errors can be quickly and easily corrected by users [4]. For each audio data of a podcast episode, this function displays not only a 1-best word sequence but also a numbered list of N-best competitive candidates for each word (not for a sentence) as shown in Figure 1. A user who finds a recognition error can simply select the correct word from the candidates. When the correct word is not shown in the candidates, the user can also correct the error by typing through the keyboard. For the generation of candidates, we adopt a *confusion network* that is a simple network representing the intermediate recognition result and is obtained by condensing a huge word graph [5].

This error correction by users can be regarded as a *social annotation* on Web 2.0. In PodCastle, the performance of both full-text search and speech recognition will be improved based on the annotation. However, in order to let users correct recognition errors easily and efficiently, the recognition results should have as high quality as possible. Hereafter, we describe the speech recognition system and methods to improve the performance of speech recognition in order to transcribe podcasts more accurately.

3. System description

Our speech recognition system aims to generate not only a 1-best word sequence but also a numbered list of N-best competitive candidates for each word. In this section, the details of each component of the baseline transcription system are presented.

3.1. Audio segmentation

In order to transcribe the continuous stream of audio data, segmentation or partitioning techniques to detect speech segments are important. In this work, a Gaussian mixture model (GMM) based approach is adopted for prior audio segmentation. As a preliminary experiment, we considered only three types of acoustic events: speech, music without speech, and other background sounds. The speech and background-sound models were trained on a Japanese speech corpus. For the music-without-speech model, we used the RWC music database (RWC-MDB-P-2001, RWC-MDB-R-2001, and RWC-MDB-J-2001) [6]. The number of the Gaussian mixtures of each model is 64. The entire audio stream is directly recognized using a conventional Viterbi decoder with each of the three GMMs in parallel. Since this approach tends to provide short segments of those events, we use an inter-class transition penalty that forces the decoding process to produce longer segments.

3.2. Acoustic model

In general, podcast audio data include various types of speech, for example, noisy speech, narrow-band speech, and speech with music. To suppress some noises in such speech segments, the ETSI advanced front-end [7] is used to estimate MFCCs in the preprocessing stage. Finally, we obtain an acoustic feature vector consisting of 39 components (12 MFCCs, normalized log energy, and their first and second differential coefficients). As for the training data of acoustic modeling, we used 600 hours of presentation speech in the Corpus of Spontaneous Japanese (CSJ) [8]. The speech recognizer uses a tied-state left-to-right context-dependent HMM with Gaussian mixture observation densities where tied states are constructed by using a top-down clustering method based on a phonetic decision tree. The cross-word triphone model has 4513 tied states and 16 Gaussians per state.

3.3. Language model

Language modeling is one of the challenging issues in recognizing the speech of podcasts. Since podcasts have various tasks and wide domains, it is very difficult to specify the topic and vocabulary to be covered in a language model beforehand. Therefore, it can be said that large-scale models are ideal to include as many words to be recognized as possible. In this work, we used 10 years of Japanese newspaper articles and transcriptions of spontaneous presentations in the CSJ as a basic text corpus for training the language model. However, regardless of how large the scale of the language models is prepared beforehand, the out-of-vocabulary issue will not be solved drastically in the recognition process. In particular, this problem is more critical for podcasts because the latest topics and words appear frequently in newly added episodes on a daily basis. To deal with this problem, daily updated text data from web news sites are also used for training of the language model. In this work, we have collected the text data for each news category from two popular web news sites (Yahoo! news and Google news) everyday. Finally, a 3-gram language model was trained on the three kinds of text corpora/resources (newspapers, the CSJ, and web news) and contains 152163 words.

3.4. Decoding

For the podcast transcription system, a multi-pass decoding strategy is used to reduce the computational costs and generate competitive candidates. Recognition is performed in three decoding passes as follows:

1. Initial hypotheses are generated using a phoneme recognizer, and the hypotheses are used for unsupervised acoustic model adaptation using the MLLR technique [9].
2. Word decoding is carried out using the adapted acoustic model. First, a word graph is generated using a lexical tree beam search with a 2-gram back-off language model. Then,

Table 1: Description of the test set

ID	Category	Length (sec.)	Read or Spontaneous	Acoustic conditions
A	commentary	192.1	spontaneous	clean
B	commentary	128.2	spontaneous	clean
C	news	1159.6	read	clean
D	news	247.3	read	music
E	chitchat	486.6	spontaneous	clean

Table 2: Perplexities (ppl.) and the number of OOV words for the test set.

	LM		LM+web	
ID	ppl.	# OOV words	ppl.	# OOV words
A	112.2	4	104.1	2
B	86.8	0	72.6	0
C	86.5	10	62.1	10
D	146.6	8	99.6	5
E	281.0	18	266.2	18

Table 3: Word error rates and network error rates for each language model.

	LM		LM+web	
ID	WER(%)	NER(%)	WER(%)	NER(%)
A	28.9	12.5	27.5	11.1
B	27.5	13.4	25.3	9.8
C	24.1	9.1	18.3	8.2
D	45.6	24.8	35.6	13.1
E	48.4	32.3	45.1	29.1

the word graph is rescored using a 3-gram language model and the word hypotheses are used for the MLLR adaptation.

3. The above mentioned word decoding is conducted again using the adapted acoustic model, and the word graph is reconstructed. Finally, the consensus decoding [5] that directly minimizes the word error rate is conducted, and a confusion network is generated.

The confusion network generated in the final process is adopted to the error correction interface as shown in Figure 1.

3.5. Experiments

The transcription performance of the baseline system was evaluated with the actual podcast speech data. For testing, we used five Japanese podcasts and selected one episode from each podcast as shown in Table 1. The test set includes three categories of podcasts: a commentary focused on the latest economics in Japan (ID: A, B), a daily news distributed by a Japanese broadcasting company (ID: C, D), and a chitchat show by a Japanese popular teen idol (ID: E).

First, we evaluated the performance of the language models. Table 2 gives the perplexity and the number of out-of-vocabulary (OOV) words of each episode. In this table, "LM" indicates the baseline language model trained on two text corpora (newspapers and the CSJ) and "LM+web" indicates the "up-to-date" language model using web news texts described in Section 3.3. Our up-to-date language model significantly reduced the perplexity of all episodes. As can be seen, it achieved significant improvements especially for daily-news speeches (C, D). This means that the up-to-date language model can capture the latest topic and vocabulary by using web-news texts.

Table 3 shows the word error rate (WER) and network error rate (NER). The NER is computed as the WER of a word sequence which best matches the correct word sequence in a confusion network. According to the improvements of the language model, the recognition performance was also improved, and the effectiveness of the use of web-news texts for language modeling was confirmed. These results show that a mechanism to keep a speech recognizer always up-to-date is needed in order to transcribe speech data like podcasts that are updated on a daily basis.

Table 4: Perplexities (ppl.) and the number of OOV words for each language model.

	LM+web		LM+web+adapt	
ID	ppl.	# OOV words	ppl.	# OOV words
A	104.1	2	103.1	2
B	72.6	0	46.9	0
C	62.1	10	61.5	10
D	99.6	5	88.9	5
E	266.2	18	193.6	18

4. Improving methods on the basis of Web 2.0

In this section, we present methods to improve speech recognition using Web 2.0 knowledge sources and contents. Recently, in Web 2.0 services and applications, a huge amount of multimedia contents and annotations are generated and accumulated by an unspecified number of users. These resources are steadily increasing and updated on a daily basis. In this work, we study the use of these useful resources for training of the speech recognizer.

4.1. Language model adaptation using RSS metadata

A podcast is distributed using RSS syndication feeds to notify users about updates. As mentioned above, the feed provides useful data such as the publishing date, titles, and accompanying text descriptions of the series and each of its episodes. In this work, we study the use of the text data in RSS for the topic adaptation of language models. The topic adaptation is conducted for each episode as follows:

1. Parse RSS "title" and "description" for nouns using a Japanese morphological analyzer *Chasen*.
2. Retrieve text documents related to the topic of the episode from the web using a text search engine. We use only nouns in the parsed text as search terms.
3. Build a web-based language model from the retrieved documents.
4. Interpolate the web-based language model with the baseline language model described in Section 3.3

In order to confirm the effectiveness of the adaptation, we evaluated the recognition performance of the test set. As a search engine, we used the *Yahoo! web search API* [12] and the number of the retrieved pages was set to a maximum of 200 per query.

The experimental results of the adaptation are summarized in Table 4 and 5. The effectiveness of the adaptation varied for each episode in the test set. For the three episodes (B, D, and E), the perplexity was reduced significantly and the recognition performance was also improved. In these episodes, the RSS feeds contained summaries of the main topics. Hence, text documents related to the topics were collected from the web. On the other hand, only slight improvements were achieved for the two episodes (A, C). This is because no useful information for the topic adaptation was given in the RSS feed.

Table 5: Word error rates and network error rates for each language model.

	LM+web		LM+web+adapt	
ID	WER(%)	NER(%)	WER(%)	NER(%)
A	27.5	11.1	27.2	11.2
B	25.3	9.8	22.9	9.0
C	18.3	8.2	18.0	8.2
D	35.6	13.1	34.5	12.9
E	45.1	29.1	42.4	27.4

4.2. Acquiring word pronunciations from the web

Podcasts cover a large range of topics including the newest proper names and buzzwords. Thus it is impossible to prepare a dictionary which includes all words (and their pronunciation) that might occur in a podcast. Many technical terms and coined words of various topics also have to be considered (e.g., PodCastle, Wikipedia, and so on) in the podcast transcription task. In this work, we try to automatically obtain their pronunciations from the web. A Japanese Web 2.0 service "*Hatena diary keyword*" [10] publishes a list of new keywords including explanations and pronunciations. The keywords of this list have been updated by anonymous users, and over 193000 keywords are registered at present. The site reports that about 300 words are added daily.

In typical Japanese LVCSR systems, the pronunciations used in the recognizer are taken from a publicly available Japanese dictionary [11] that includes various information such as grammar, parts of speech, and pronunciations of words. Although the dictionary contains a large amount of general words, it is insufficient when dealing with podcasts as mentioned above. In fact, the pronunciations were not obtained from the dictionary for 11.5% (17438/152163) of words in our up-to-date language model. By using the pronunciations of Hatena diary keyword, the correct pronunciations were obtained for 22.9% (3997/17438) of the words. Unfortunately, we could not confirm the effectiveness on the recognition performance, because these words do not occur in the test set. However, since some of these words are very popular, we believe that this approach will improve the performance of the speech recognition system in general.

4.3. Acoustic model training based on error corrections by users

In PodCastle, the manual annotations for podcasts are basically the correction of speech recognition errors. Although some users may thoroughly correct all recognition errors, most users will correct only errors related to keywords or key sentences according to their interests. Therefore, the use of error correction results to improve the speech recognizer is a considerable challenge. As the first step of this challenge, we use error correction results for training the acoustic model. In a podcast, the acoustic conditions tend to be similar throughout all episodes, such as speakers, recording conditions, and the level of background music. We therefore first trained the acoustic model by using one episode of each podcast and then evaluated the performance improvement by using the other episodes in the same podcast. For each of the five podcasts, one episode where errors were corrected by users was used for training. However, these error corrections were not necessarily carried out perfectly. For example, only about 50% of the recognition errors were corrected for the episode of podcast E. To estimate the HMM parameters, we used the MLLR-MAP method: i.e., the MLLR transformed parameters were used as the priors for MAP estimation. Table 6 shows the effectiveness of the acoustic training. Although the amount of speech data was limited and the corrected transcriptions used as the training data were not perfect, the recognition performance was improved overall. In particular, a larger improvement was achieved for podcast D that has an adversed acoustic condition. From these results, we confirmed that the podcast-dependent training of the acoustic model was effective.

Table 6: Performance of MLLR-MAP training.

	before training		after training	
ID	WER(%)	NER(%)	WER(%)	NER(%)
A	27.2	11.2	25.1	9.4
B	22.9	9.0	19.3	8.5
C	18.0	8.2	16.1	7.9
D	34.5	12.9	27.8	9.1
E	42.4	27.4	40.1	25.1

5. Conclusions

We have presented speech recognition techniques that enable a new web service *"PodCastle"* for searching, reading, and annotating podcasts. To overcome the difficulties in transcribing podcasts, we proposed methods for the language model updating using web news texts, the language model adaptation using RSS metadata, and the pronunciation acquisition using a Web 2.0 service. Furthermore, the results of our preliminary experiments show that the acoustic model can be improved by using transcriptions corrected by users.

In future work, we plan to further investigate the effectiveness of the methods presented in this paper in larger-scale experiments. We will also study other training techniques for the speech recognizer so as to make best use of user corrections. At present, the recognition performance degrades drastically for very complex podcasts such as those including human-human conversations, speech with loud background music, and recognition of very emotional speech. We will tackle these issues using the advantages of PodCastle and the Web 2.0 framework.

6. References

[1] D. Miller, *et al.*, "BBN at TREC7: Using hidden Markov models for information retrieval," in *Proc. of TREC-7*, pp.133–142, 1999.

[2] J.-M. V. Thong, *et al.*, "Speechbot: An experimental speech-based search engine for multimedia content on the web," *IEEE Transactions on Multimedia*, vol.4, no. 1, pp. 88–96, 2002.

[3] M. Goto, *et al.*, "PodCastle: A Web 2.0 Approach to Speech Recognition Research," in *Proc. of Interspeech 2007*, 2007.

[4] J. Ogata and M. Goto, "Speech Repair: Quick error correction just by using selection operation for speech input interfaces," in *Proc. of Eurospeech 2005*, pp. 133–136, 2005.

[5] L. Mangu, *et al.*, "Finding consensus in speech recognition: word error minimization and other applications of confusion network," *Computer Speech and Language*, vol.14, no.4, pp.373-400, 2000.

[6] M.Goto, "Development of the RWC music database," in *Proc. of ICA 2004*, pp.I-553–556, 2004.

[7] ETSI ES 202 050 v1.1.1 STQ

[8] T.Kawahara, *et al.*, "Benchmark test for speech recognition using the corpus of spontaneous Japanese," in *Proc. SSPR 2003*, pp.135–138, 2003.

[9] C.L. Leggetter and P.C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech and Language*, vol.9, pp.171–185, 1995.

[10] Hatena diary keyword, http://d.hatena.ne.jp/keyword

[11] ipadic, http://chasen.naist.jp/stable/ipadic

[12] Yahoo! web search API, http://developer.yahoo.co.jp/search