Unsupervised Learning from Users' Error Correction in Speech Dictation

Dong Yu, Mei-Yuh Hwang[1], Peter Mau, Alex Acero, Li Deng

Speech Technology Group
Microsoft Research / Redmond, USA
{dongyu,petermau,alexac,deng}@microsoft.com

Abstract

We propose an approach to adapting automatic speech recognition systems used in dictation systems through unsupervised learning from users' error correction. Three steps are involved in the adaptation: 1) infer whether the user is correcting a speech recognition error or simply editing the text, 2) infer what the most possible cause of the error is, and 3) adapt the system accordingly. To adapt the system effectively, we introduce an enhanced two-pass pronunciation learning algorithm that utilizes the output from both an n-gram phoneme recognizer and a Letter-to-Sound component. Our experiments show that we can obtain greater than 10% relative word error rate reduction using the approaches we proposed. Learning new words gives the largest performance gain while adapting pronunciations and using a cache language model also produce a small gain.

1. Introduction

We have seen significant progress in advancing Automatic Speech Recognition (ASR) technologies in the past several decades. However, ASR systems are not widely adopted today. Among all the issues that prevent users from using ASR systems, recognition accuracy is still the most important factor [1].

It is well known that an ASR system adapted to a specific user performs better than an out-of-box system which comes with a speaker independent Acoustic Model (AM), Language Model (LM), and dictionary. For this reason, it is recommended for users to go through several AM adaptation sessions before using dictation system. While this pre-usage supervised adaptation approach helps to improve the overall performance, it's not sufficient by its own due to the fact that there are more elements to be adapted than the AM alone. For example, the default dictionary does not include some of the words frequently used by a specific user. Examples of these Out Of Vocabulary (OOV) words are project names, acronyms, and foreign names. Adapting the lexicon would recover on average 1.2 times as many errors as OOV words removed [2]. Another element that requires adaptation is word pronunciations. This is especially true when a word is a foreign name or the user is not a native speaker [3, 4]. Furthermore, different users may pronounce some of the words differently.

To further improve the recognition accuracy, we explore additional ways to adapt the ASR system. Specifically, we propose an approach to adapting the ASR system through unsupervised learning from users' error corrections.

It is easy to see that users' corrections contain useful information to adapt the ASR system. However, using such information without supervised information is not easy. We pursue the unsupervised adaptation approach because most dictation users are not ASR savvy and don't know how to "teach" the ASR system to perform better. For example, they usually don't know whether they should add a new word to the lexicon, or provide a pronunciation even after a suggestion is made to them. Another reason is that having the system request specific information from users would slow them down and probably annoy them.

In this paper, we introduce our new approach to tackling the difficulties encountered when adapting the ASR system with users' corrections. We propose an architecture that would infer the likelihood that the user is making a correction instead of changing his/her mind, automatically identify the most possible cause (e.g. OOV, pronunciation, LM) of the ASR error, and adjust the ASR system accordingly. To correctly learn the pronunciation, we propose a pronunciation recognizer that picks the best phoneme sequence based on the result of a standard n-gram phoneme recognizer and the pronunciations generated by the Letter-to-Sound (LTS) component.

The rest of the paper is organized as follows. In section 2, we introduce the strategy used by our learning from correction (LFC) system, including the algorithm used to infer whether the user is making a correction or changing his/her mind. In section 3, we describe how pronunciations can be reliably learned from the corrections. We present experimental result in section 4, and conclusions in section 5.

2. Learning from Corrections: Strategy

The strategy of our LFC approach is summarized as the flow chart depicted in Fig. 1. The system first detects whether the user has changed the dictated text. If yes, the system will then infer whether the user is correcting a speech recognition error or simply editing correctly recognized text. If it's not likely to be an ASR correction, nothing will be learned. Otherwise, the system will check whether the error is likely caused by OOV words. If yes, the new words are added into the lexicon. Otherwise, the system continues to see whether the error is likely caused by an incorrect pronunciation. If yes, it will learn the pronunciation and determine whether that pronunciation should be added into the lexicon. This process goes on until we don't have anything to learn. To prevent over learning, we only adjust the most important factor for each correction.

[1] Mei-Yuh Hwang (mhwang@ee.washington.edu) is currently with Department of Electrical Engineering, University of Washington, Seattle, USA.

2.1. Infer User's Intention

When a user edits the dictated text, the user may in fact not make an ASR correction. If the LFC module blindly learns from all the edits the user has made, it will gradually corrupt the ASR system. For this reason, the first task of our LFC module is to infer how likely a specific editing is really an error correction.

Users make corrections in different ways: selecting from an alternate list, re-dictating, or editing with the keyboard. The difficulty comes from the fact that users may use these same ways when they are not making a correction, e.g., polishing the documents, or changing his/her mind during dictation. The LFC module infers whether the user is making an error correction based on the text changed From (F), the text changed To (T), the acoustic Audio (A), and the Editing approach (E).

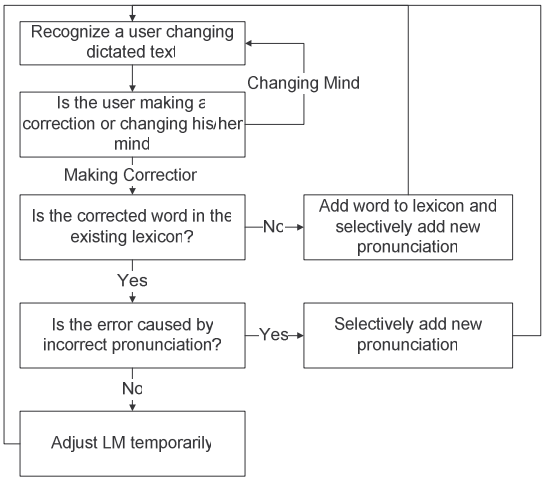

*Fig. 1: F*low chart of the LFC approach

The decision is made based on the likelihood ratio *r(F,T,A)* defined in Eq. 1.

$$r(F,T,A) = P(T|A)/P(F|A)$$

$$= P(A|T)P(T)/(P(A|F)P(F)) \quad (1)$$

P(T) and *P(F)* can be directly calculated through language model. The AM score of the original text *P(A|F)* can be directly retrieved from the original recognition result. To calculate the AM score of the updated text against the original audio (*P(A|T)*), we do a forced alignment against the original audio using the updated text and the context words: the previous one word and the following one word, if available. For example, if the original recognition result is: "This is a text .\period" and the user corrected "text" to be "test", the phrase "a test .\period" is then used as the reference transcription to align the original audio and label the most possible boundaries of the corrected words.

The LFC component decides that the editing is a correction if

$$r(F,T,A) > \theta_E \quad (2)$$

where θ_E has different values for different editing approaches and can be estimated using training data. Adjusting the thresholds θ_E would control how aggressive the LFC module adapts the ASR system. It's impossible to eliminate all instances that the user is not making a correction with this intention inference stage, so some additional safeguards are needed in the later stages to prevent incorrect adaptation.

2.2. Adding New Words

Checking whether the error is caused by OOV words is easy and reliable. We first convert the corrected phrase into a normalized form. We then examine each word in the normalized text to see whether it exists in the lexicon with a table lookup. A word is an OOV word if it's not in the lexicon. To adapt the system, we add that word into the lexicon and boost its LM unigram score based on the following formula:

$$P(w)=\lambda_t P_0+(1-\lambda_t)P_w \quad (3)$$

where λ_t is a interpolation weight that equals 1 initially and exponentially decays to 0 over time. P_w is the estimated unigram score based on the counts. P_0 is a fixed value so that the score is high enough (but not too high) originally that this word may survive in the pruning process. We do this since the occurrence of histories has the property of locality [5, 6].

2.3. Adjusting LM

If it's identified that the cause of the error is likely to be the LM, we consider adjusting it. LM is usually trained with millions of words. Adjusting it based on a few occurrences is not desirable. For this reason, we only boost the LM score in the duration of a session following the work of [6].

For example, if "wave two" is misrecognized as "wave too" and gets corrected by the user, the system automatically boosts the bi-gram *P(two|wave)* so that the combined AM and LM score of "wave two" is larger than that of "wave too" with a marginal value for that speech fragment during this session.

2.4. Learn Pronunciations

Learning pronunciations automatically from users' corrections is the most difficult part due to three reasons:

- We need to find a way to correctly align the audio based on the corrected phrase so that the correct pronunciation can be learned.
- We need to have a good pronunciation recognizer to learn the correct pronunciations from the audio and the associated text.
- We need to devise a way to determine whether the pronunciation learned for a word is reliable and whether it should be added into the lexicon.

Because pronunciation learning involves higher complexity than other adaptation components in this work, we devote section 3 to resolving the above issues.

3. Pronunciation Learning

Learning pronunciation is very important for words such as acronyms and foreign names. Usually, pronunciations for these words are either provided by LTS or learned through an n-gram phoneme recognizer.

LTS rules tend to do a good job on regular words, i.e., the words that are more predictable in a particular language. If a word's pronunciation is hardly related to how it is spelled (such as a foreign name) or if the user has a strong accent (e.g., a Japanese user typically pronounces the word "mail" as "m ey l uw"), LTS would fail to provide a good pronunciation. The n-gram phoneme recognizer, on the other hand, tries to capture any sequence of phonemes but the pronunciation learned might be biased by the n-gram trained from the known words.

The problem is particularly severe for those words that are mixed with regular words and foreign words or acronyms. For example, an LTS system is likely to produce the following pronunciation for the word "voicexml":

v oy s eh k s m ax l

A recognizer using the all-phone network, however, is likely to do a worse job at the beginning but much better job at the end:

ow s eh k s eh m eh l

Choosing either pronunciation would be so wrong that the word "voicexml" can't be recognized even after it's added into the lexicon.

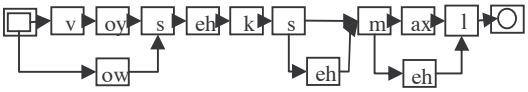

Figure 2: Phoneme graph for the word "voicexml": This graph is constructed from the phoneme sequences generated from both LTS and the n-gram phoneme recognizer

The optimal pronunciation θ given acoustics A and transcription T is expressed by

$$\theta = \arg\max_{\theta} p(\theta \mid A,T) = \arg\max_{\theta} p(A \mid \theta)\, p(\theta \mid T) \quad (4)$$

Initially $p(\theta \mid T)$ is given by LTS, which often generates several pronunciations. Since LTS may not generate the correct pronunciation for words such as acronyms and foreign names, we want to relax such component. One possibility is smoothing the LTS contribution with an all-phone network which leads to a less optimal solution.

Our solution is to construct a phonetic graph from all the sources (including LTS generated pronunciations and phonemes recognized by the n-gram phoneme recognizer) and then to rescore and pick the best path based on the acoustic sample. In other words,

$$p(\theta \mid T) = \begin{cases} 1/N, & \theta \text{ is a path in the graph} \\ 0, & \text{othwewise} \end{cases} \quad (5)$$

where N is the total number of paths in the graph. The solution is summarized as:

1. Use the n-gram phoneme recognizer to search for the best phoneme sequence from the acoustic sample.
2. Use all lexicons available and the LTS to provide a list of possible pronunciations with the given spelling.
3. Construct a phoneme graph based on the pronunciations collected from steps 1 and 2.
4. Rescore the phoneme graph with the acoustic sample and output the best phonetic pronunciation.

For example, "voicexml" has the phoneme graph shown in Figure 2. The correct pronunciation is a mixture of phonemes from all sources constrained by the graph.

Having a good pronunciation recognizer is not the end of the story. Two other factors may affect the result of the pronunciations recognized. The first factor is that the user's editing is not really a correction but the LFC component believes it is. When this condition occurs, the pronunciations recognized might be completely wrong. The second factor is the possible wrong alignment due to the incorrect original pronunciations. When this situation occurs, the result from the pronunciation recognizer usually has incorrect first and/or last phoneme. We partially fix incorrect pronunciations generated due to this factor by comparing the first phoneme in the result with the last phoneme in the previous word's pronunciations, and the last phoneme in the result with the first phoneme in the posterior word's pronunciations.

To prevent those incorrect pronunciations from being added into the lexicon, we determine whether the new pronunciation should be added into the lexicon based on a confidence score generated from the distance between the new pronunciation and existing pronunciations and the frequency of the new pronunciation. The new pronunciation is added if the confidence is larger than a threshold, the AM score change with the new pronunciation is larger than a threshold, and the new pronunciation happens at least twice.

4. Evaluation

To evaluate our LFC approach, we conducted two experiments: an offline experiment and an online experiment.

In the offline experiment, we collected emails sent by 4 users and asked them to read them in the order they were composed and we recorded the audio. Each user spoke 2000-4000 words. We then used an offline testing tool to simulate the correction process. When a recognition error occurred, we recorded that as an error. At the same time, the tool provided the corrected text and our system learned from the correction. This process continues until all audio files are processed from each user.

Table 1: Offline experiment result on WER (%)

WER (%)	User1	User2	User3	User4	Average
Baseline	16.7	18.1	21.3	17.3	18.4
New Word (NW)-Inc	14.8	17.5	20.6	13.3	16.6
NW-Aft	12.2	15.7	18.6	10.6	14.3
NW+Pron-Inc	14.7	17.4	20.6	12.7	16.4
NW+Pron-Aft	11.2	15.5	18.6	9.8	13.8
NW+Pron+LM-Inc	14.7	17.1	20.6	12.8	16.3
NW+Pron+LM-Aft	9.8	12.8	16.7	8.6	12.0

Table 1 summarizes the result of the offline experiment. Some of the pronunciations learned are listed in Table 2. *Base line* stands for the ASR system without LFC. *NW* means adding New Words, *Pron* means adapting new pronunciations, and *LM* means using the cache LM described in Section 2.3. The suffix *Inc* denotes incremental testing that each error is counted and the correction information is used to adapt the system for future audio segments. The suffix *Aft*

means testing without learning after the same test set has been used for an incremental testing (the ASR is already updated). The actual error rate should be the *Inc* value if the test set size was large. Given that our test size is small, we hypothesize the actual error rate is in between the *Inc* and the *Aft* values and is closer to the *Aft* values, since the most of what the system will learn are user specific and can be captured after some finite period (except the LM cache). This is especially true for pronunciations since we add a pronunciation into the lexicon after at least seeing it for three times and so the pronunciation learning would only take effect when the same pronunciation is shown the fourth or more times.

From Table 1, we can see that we get at least 11% relative WER reduction by introducing the LFC. Adding OOV words gives us the largest gain. Adapting pronunciations helps a little and using cache LM gives us an insignificant gain. Also note that LFC works better for some users (user1 and user4) but is not very effective for some other users (user2 and user3).

Table 2: Examples of the pronunciations learned

Spelling	Lexicon or LTS Pronunciation	Learned Pronunciation
sgstudio	z g s t uw d iy ow	eh s jh iy s t uw d iy ow
mste	m s t	ax m eh s t iy iy
z	z iy	s eh d
ASI	ey z iy	eh s ay
smex	s m eh k s	eh s eh m iy eh k s
HMIHY	hh m iy hh iy	hh aw m ey ay hh eh l p y uw

For the online experiment, we asked users to install our experimental system and use it normally. The experimental system automatically detects whether the user is making correction, learns from corrections, and reports back the dictated text and audio. Fig. 3 shows the recognition accuracy over time for seven users that used the system long enough.

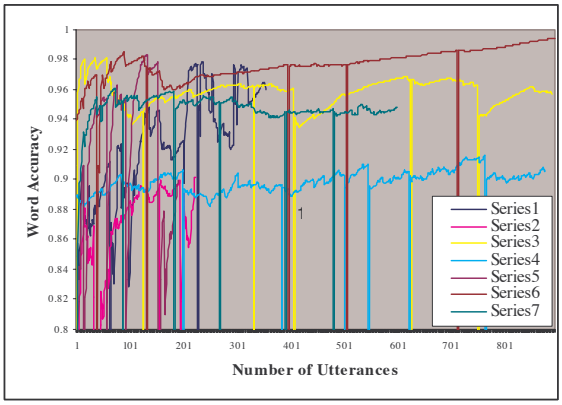

Fig. 3. Online experiments: vertical lines indicate the clearance of the LM cache

The X-axis in Figure 3 indicates the number of utterances (five words per utterance on average) recognized by the ASR system. The Y-axis is the 1000 words moving average word accuracy. The vertical lines in the figure indicate the end of an old session and start of a new session at which time the cached new word unigram and cached LM are cleared. From the figure, we can see that the recognition accuracy increases over time within each session and LFC does provide some benefits. We also observe that the cached word unigram is important to improving the ASR performance.

5. Conclusions and Future Work

We introduced our recent work in unsupervised learning from users' error correction. We described the learning strategy, users' intention inference algorithm, and the improved pronunciation recognizer. We showed that with LFC, we can reduce the WER incrementally by more than 10% without intervention from users. With all the elements adapted, adding new words is very reliable and gives us the best gain. It appears that giving the new words a boosted unigram is very important to get the gain we expect.

Our LFC system can be further improved in the following two areas.

First, in the current system, we only adapt the vocabulary, pronunciations, and the LM. We perceive that users' correction information can also be used to adapt the AM. Today's unsupervised AM adaptation occasionally decreases the accuracy as time progresses. With users' correction information, we can improve unsupervised AM adaptation by using only those uncorrected sentences and/or edited phrases that are highly likely corrections.

Second, in the current system, we did not use the system's overall recognition accuracy as a guide when determining whether we want to learn a new pronunciation or adjust the LM. For example, if the system's average WER is 10% and a word's average WER is 9%, it might not be desirable to adapt the AM or LM for this word. However, if a word's average WER is 90%, it's a good indication that we should learn something to improve the accuracy for this word.

6. Acknowledgements

We wish to thank the members of the speech group at Microsoft Research in Redmond for valuable discussions, and anonymous reviewers for great comments and suggestions.

7. References

[1] L. Deng, and X. Huang, "*Challenges in Adopting Speech Recognition*", Communications of ACM, vol. 47, No. 1, pp69-75, Jan 2004.

[2] L. Lamel and G. Adda, "*On Designing Pronunciation Lexicons for Large Vocabulary,Continuous Speech Recognition*", Proc. International Conference on Spoken Language Processing (ICSLP'96), pp6-9, 1996.

[3] Dong Yu, Kuansan Wang, Milind Mahajan, Peter Mau, and Alex Acero, "*Improved Name Recognition With User Modeling*", Proc. Eurospeech'03, pp1229-1232, 2003.

[4] F. Beaufays, A. Sankar, S. Williams, and M. Weintraub, "*Learning Linguistically Valid Pronunciations from Acoustic Data*", Proc. Eurospeech'03, pp2593-2596, 2003.

[5] J. Wu, "*Maximum Entropy Language Modeling with Non-Local Dependencies*", Ph.D. dissertation, Johns Hopkins University, 2002.

[6] R. Kuhn and R. De Mori, "*A cache-based natural language model for speech recognition*", IEEE Trans. on Pattern Analysis and Machine Intelligence, 12(3), pp. 570-583, 1990.