Efficient Speech Transcription Through Respeaking

Master's Thesis

Matthias Sperber

November 5, 2012

Supervisors: Dr. Graham Neubig
Dr. Christian Fügen
Prof. Dr. Satoshi Nakamura
Prof. Dr. Alexander Waibel

Karlsruhe Institute of Technology
Department of Computer Science
Institute for Anthropomatics
Prof. Dr. Alexander Waibel

Matthias Sperber
Münchbuschweg 30b
67069 Ludwigshafen

Ludwigshafen, den 5. November 2012

Erklärung

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

(Matthias Sperber)

Efficient Speech Transcription Through Respeaking

Matthias Sperber

November 5, 2012

Abstract

In this thesis, we propose a method for efficient speech transcription through respeaking. Speech is segmented into smaller utterances using an initial automatic transcript. Respeaking is performed segment by segment, while confidence filtering helps save supervision effort. We conduct detailed experiments comparing speaking vs. typing, the effect of sequential vs. confidence-ordered supervision, and respeaking word error rate on correction efficiency. Our results demonstrate that the proposed method can not only outperform typing in terms of correction efficiency, but is also much less demanding for the respeakers than traditional methods, and consequently helps keep costs down.

Effizientes Transkribieren von Sprache durch Nachsprechen

Matthias Sperber

5. November 2012

Zusammenfassung

Diese Arbeit untersucht, wie gesprochene Sprache durch Nachsprechen effizient transkribiert werden kann. Eine Sprachaufnahme wird anhand eines initialen, automatischen Transkripts in kleinere Segmente unterteilt. Mit einem Konfidenzfilter werden diejenigen Segmente identifiziert, deren initiales Transkript einer Korrektur bedarf. Die zu korrigierenden Segmente werden dann nachgesprochen, ein Spracherkenner erzeugt ein neues Transkript. Dieses ist aufgrund eines sprecher-adaptierten Spracherkenners sowie günstigeren Aufnahmebedingungen in der Regel deutlich besser als das initiale Transkript, gleichzeitig ist diese Korrektur-Methode deutlich schneller als das Eintippen über die Tastatur. Schließlich werden die initiale und die durch Nachsprechen erzeugte Hypothese noch kombiniert, sodass idealerweise aus beiden Hypothesen jeweils genau die korrekten Wörter für das finale Transkript gewählt werden.

Im Unterschied zur weit verbreiteten Methode, in der eine Sprachaufnahme nicht segmentiert wird, sondern als ganzes ohne Pause nachzusprechen ist, ist unser Ansatz weitaus einfacher durchzuführen und erfordert kein spezielles Training. Dadurch ist unsere Methode kostengünstig einsetzbar.

Mithilfe eines effizienten, eigens zu diesem Zweck entwickelten Tools führen wir detaillierte Experimente durch, um die Korrektur durch Nachsprechen oder Tippen zu vergleichen, und den Einfluss der Wortfehlerrate beim Nachsprechen auf die Effizienz der Korrektur zu analysieren. Weiterhin vergleichen wir sequentielle Korrektur und Korrektur in Reihenfolge der Konfidenzen. Unsere Ergebnisse zeigen, dass unsere Methode des Nachsprechens effizienter als Tippen sein kann, wobei dies von Sprecher zu Sprecher unterschiedlich ist. Auch der Einsatz der Konfidenzen sowie die Hypothesen-Kombination bewirken deutliche Effizienz-Steigerungen.

Schließlich zeigen wir das weitere Steigerungspotenzial der Effizienz durch bessere Sprecher mithilfe von simulierten Experimenten auf, und diskutieren die Möglichkeit, von Segment zu Segment zu entscheiden, ob die Korrektur durch Nachsprechen oder Tippen geschehen sollte.

Acknowledgements

I would like to thank my supervisors Christian Fügen and Graham Neubig for the advice and the many fruitful discussions that helped me carry out this project. Many thanks to Professor Alexander Waibel for providing me with the great opportunity of working on my thesis in cooperation with the Nara Institute of Science and Technology, as well as his supervision and his interest and support for my research. My sincere thanks go to Professor Satoshi Nakamura who welcomed me at his laboratory and very generously provided for everything I needed to accomplish my goals. I would furthermore like to thank Sakriani Sakti and Tomoki Toda for their helpful suggestions.

I am very grateful to all my fellow students, especially Michael Heck and Keigo Kubo. It was fun working together as a team. Many thanks also to Manami Matsuda for her big effort in supporting my stay in Japan, and Hiroaki Shimizu who let me stay at his house when I had no place else. I am grateful to all the other members of the laboratory who welcomed me as part of the group and made me feel quite at home in Japan.

Last but not least, my sincere thanks go to my dear fiancée and to my family for their love, support, and encouragement, making all of this possible in the first place.

Contents

Chapter 1

Introduction

While the transcription of speech is a necessity for an increasing number of applications, often quality requirements are high and cannot be met even by state-of-the-art automatic speech recognition (ASR) technology. On the other hand, manual transcriptions are very expensive. The combination of human skills and speech technology can help ameliorate these problems by providing a good trade-off between transcription quality and cost. We therefore formulate the goal of enabling efficient transcription of speech by combining ASR technology and manpower.

The advantages of ASR technology are its speed, low costs, and the ability to work without ever getting tired. Human transcribers, on the other hand, are expensive and suffer from fatigue, but unlike ASR technology can obtain a real understanding of spoken language, are very strong at inferring from context, as well as at segmenting the individual words, or words and background noise. As a result, humans are much better at choosing the correct among several plausible hypotheses, perform effective error recovery, and are less influenced by other disturbing factors.

To work towards this goal of effectively combining the strengths of both, we investigate the efficiency of *respeaking* as a method for speech transcription. We define respeaking as a person simultaneously repeating the same speech that another person is uttering. The respoken utterances can then automatically be transcribed using ASR. Since the data entry rate of speech is much higher than that of typing on a keyboard [Moore et al., 2004], the transcription task can be considerably sped up by using respeaking. While the quality of the resulting transcript depends on the recognition accuracy, this accuracy can be expected to be much better than for recognizing the original speaker, since we can adapt the system towards the particular respeaker, and moreover assume better recording quality. In this way, respeaking can provide a good trade-off between transcription effort and accuracy.

The television broadcast industry has already identified respeaking as a promising way to create live-subtitles for their programs. In fact, respeaking has now largely replaced the previously predominant method of stenography for this task [Evans, 2003, Prazak et al., 2012]. However, respeaking in real-time is very demanding for the speaker, requires extensive training, and often results in a summary rather than a faithful transcript. We attempt to overcome these limitations by dividing the speech into smaller segments, based on the initial transcript. Instead of respeaking all speech to be recognized, as in traditional methods, we select only some of the segments to be respoken based on confidence measure estimates. Further improvement is achieved by combining the hypotheses of the respeaker and the original speaker. The presented approach is not designed for use in a live setting, but rather to be applied as a post-processing step. It is "friendly" to the respeaker, as he no longer has to hurry to keep up with the original speaker. Consequently, the resulting transcripts stay closer to the original wording, and respeaking requires less training and can be executed for longer periods of time without a break than with traditional methods. We present results from experiments by three respeakers, as well as a simulation. The results demonstrate that our method is fast and more efficient than transcribing via typing or more traditional respeaking techniques, provided the speaker has a good performance in terms of recognition word error rate (WER).

Chapter 2

Background

In this chapter, we present background information that will be helpful for a thorough understanding of the succeeding chapters. We first give an overview over modern approaches to ASR, and then review some of the literature related to respeaking and speech transcription.

2.1 Automatic Speech Recognition

Automatic speech recognition is the task of automatically transcribing some given speech. This is helpful in many ways, such as enabling spoken commands, indexing and archiving, and further processing of the transcript such as by a machine translation system. In many of these use cases, the manual creation of transcripts is much too costly, leaving only automatic or semi-automatic approaches as feasible options. Yet, ASR is a highly complex undertaking, greatly complicated by the variability of speech, which can be caused by different speakers, speaking styles, background noise, or for no apparent reason. This section gives a very basic overview of how modern approaches face these challenges.

Formally speaking, the task of a speech recognizer is to calculate $\hat{W} = \text{argmax}_W P(W|X)$, thus determining the most probable word sequence $\hat{W}$ given some speech X as input. Since it is difficult to estimate this probability directly, it is commonly reversed by using Bayes' theorem, yielding

$$\begin{aligned} \hat{W} = \underset{W}{\text{argmax}} P(W|X) &= \underset{W}{\text{argmax}} \frac{P(X|W) \cdot P(W)}{P(X)} \\ &= \underset{W}{\text{argmax}} P(X|W) \cdot P(W). \end{aligned} \tag{2.1}$$

From a practical point of view, at its core, an ASR system comprises a decoder that tries to find the most probable of all possible word sequences.

Besides a good decoding method, one must further provide a feature representation X for the given audio data, an acoustic model estimating the likelihood of the word sequence $P(X|W)$, and a language model estimating the prior $P(W)$.

A good feature representation is one that contains as little redundant information as possible, keeping only what is important to recognize the uttered speech. A simple standard approach involves A/D-conversion of the audio signal, windowing, a discrete Fourier transform, and computation of the magnitude spectrum. This spectrum can then be represented compactly using filter banks, where using the mel-scale helps achieve a feature representation that resembles human pitch perception. Other methods aim at extracting even more significant information while keeping the number of features low.

The purpose of the acoustic model is then to model the relation between said features and spoken words. Since it is not immediately clear how to model acoustic features for a word as a sequence of letters, an intermediate phone-based representation can be obtained from a pronunciation dictionary. The acoustic model then follows a probabilistic approach to account for the variability in speech. In its basic form, every word is represented by a hidden Markov model. The states correspond to phones that may be traversed in temporal order. The phones are unobserved (hidden), but are modeled to emit the observed acoustic features, with emission probabilities often modeled as Gaussian mixture densities. Estimation of the model parameters is performed by using a corpus of example utterances along with their manually created transcripts.

The language model estimates the prior probability of a sequence of words, that is, without considering the acoustic observation. Intuitively, it scores a word as to how likely it is to appear in the current context, e.g. by considering the syntax, semantics, discourse information, and so on. A simple but surprisingly effective approach to language modeling is based on n-grams in the following manner:

$$P(W) = P(w_1 \dots w_k) = P(w_1) \cdot \prod_{i=2}^{k} P(w_i | w_1 \dots w_{i-1}) \tag{2.2}$$

$$\approx P(w_1) \dots P(w_{n-1} | w_1 \dots w_{n-2}) \cdot \prod_{i=n}^{k} P(w_i | w_{i-n+1} \dots w_{i-1}) \tag{2.3}$$

Here, the joint probability of a word sequence is broken up using the product rule, and then simplified using the assumptions that the conditional

probability of every word only depends on a short history of $n-1$ words. While incorrect, this assumption along with the use of interpolation and back-off methods makes it possible to estimate robust conditional probabilities on available text corpora.

Finally, the decoding process, also called the search, attempts to find the most likely hypothesis given a speech input and the previously described statistical models. Since an exhaustive search over all possible word combinations is computationally infeasible, efficient heuristics must be applied. A common data structure to support decoding is a word lattice. The lattice can for example be created by a beam search that keeps only promising hypotheses, in order to constrain the search space. Often, words are grouped into phone prefix trees according to pronunciation to avoid redundant computations. Another strategy of saving computation time is multi-pass decoding, in which only a roughly constrained search space is laid out in a first pass, which then forms the basis for a more refined search in a second pass.

When all is said and done, good evaluation metrics are needed to enable comparison of ASR approaches. The standard measure is the word error rate (WER), given by

$$\text{WER} = \frac{S + D + I}{N} \tag{2.4}$$

where S, D, and I denote the minimal number of edits (specifically, the number of substitutions, deletions, and insertions) necessary to transform the reference transcript into the recognized hypothesis. N is the total number of words in the reference.

2.2 Acoustic Model Adaptation

When using an ASR system to recognize speech, the performance often suffers from the fact that the particular speaker's voice and acoustic environment differ substantially from that of the training data. The goal of acoustic model adaptation is to overcome this mismatch by adapting the acoustic models to better fit the current situation. Since the amount of available speaker-specific data is usually rather small, a full training of new acoustic models for this speaker is not a good option. Instead, the common approach is to train an initial speaker-independent recognition system, and then adapt the system by some transformation for either the model parameters or the feature vectors, estimated on the adaptation data. Acoustic model adaptation can be supervised or unsupervised, and be applied incrementally or in batch mode. For respeaker enrollment, a method for supervised batch adaptation is appro-

priate. Depending on the amount of data, popular adaptation methods are based on maximum a posteriori (MAP) or maximum likelihood estimations. In this thesis, we choose the latter (see Section 3.2).

2.3 Transcription

Transcription is the task of putting spoken language into a text representation. The ultimate goal of automatic speech recognition is to enable fully automated transcription. However, perfect accuracy is not in sight for ASR. Moreover, modern speech recognition relies on the availability of speech transcripts as training material. The question of how to efficiently create speech transcripts with the help of human transcribers is therefore an important research topic.

Several suggestions for efficient transcription user interfaces and workflows have been proposed in recent years. In [Rodriguez et al., 2007], the goal is to create a perfect transcript, using an automatic transcript as a starting point. Supervision is performed as a number of alternating verification and correction steps. A user reads, and thereby verifies the automatic transcription, until he finds an error, which he corrects. After correction, the transcript corrected so far is used to improve the acoustic and language model. The remaining speech is then re-transcribed, possibly improving the results and reducing user effort. Although in this thesis no acoustic or language model adaptation is performed based on the user corrections, it would be reasonable to do so in the future. [Sanchez-Cortina et al., 2012] take on a different approach, in which a user specifies a tolerable maximum word error rate, and is then requested to correct the transcription for a series of words, until an estimate of the word error rate reaches the predefined threshold. Words are presented in the order of lowest confidence, to minimize the supervision effort. This second approach is similar to the one investigated in this thesis, although only typing is considered as an input method, and no segmentation is used.

[Kubat et al., 2007] present a streamlined interface that can handle large amounts of data, including multiple audio and video channels. The channels are filtered for noisy and silent parts, segmented, and presented to the human for annotation. [Huggins-Daines and Rudnicky, 2008] propose multimodal input for mobile devices. Speech is used as a first input step, after which the user can select the correct one among competing hypotheses for each word, using touch gestures. In [Luz et al., 2008], a combination of previously described methods is presented, along with possibilities to correct false word segmentations, and dynamically adapting the recognition vocabulary. The

authors further present a 3D visualization of a decoding lattice that can be used in real time. All of these ideas are applicable in our respeaking scenario as well and should be considered in future work.

In [Moore et al., 2004], the observation is made that for speech as an input method, after accounting for error correction, the effective number of words per minute attainable with speech recognition drops to within the range attainable by an average typist. Our experiments confirm this observation to hold for transcription efficiency only in a pessimistic scenario with an underperforming respeaker. For transcription, [Hsu and Glass, 2009] estimate that using efficient user interfaces, 300 words can be transcribed within 15 minutes, which is roughly equal to just under 2 minutes of spontaneous English speech [Romero-Fresco, 2009]. For comparison, the transcription tool developed as part of this thesis supports speeds about three times faster for typing, and up to five times faster for respeaking.

In contrast to the methods described above, creating transcriptions through respeaking is an approach to optimize transcription efficiency beyond that of typing. Respeaking requires less effort, but is also less accurate. For example, it generally fails for words outside of a pre-defined vocabulary. The level of faithfulness depends on the respeaker's attitude, although including non-speech events would require dedicated speech commands. Creating a perfect transcript through respeaking is only possible when using keyboard input as a back-off strategy.

2.4 Respeaking in the Television Industry

The television industry has already identified respeaking as a useful tool for their live captioning services [Evans, 2003]. Captions are transcripts of the speech contained in video material, and, unlike subtitles, are usually produced in the same language as the speech. Captioning includes transcription of the speech and other acoustic content, aligning it with the original material, and presenting it simultaneously. It is helpful to enable the deaf or hard-of-hearing to consume video material, for the silent presentation thereof, and for archiving purposes. Since many of these programs, such as news broadcasts, are presented live, creating live captions for these programs requires highly efficient methods. Moreover, the European Parliament made it mandatory for all public-service TV programs in the EU to be presented with captions, further increasing the need for efficient captioning methods [Romero-Fresco, 2009].

There are basically three approaches to captioning [Romero-Fresco, 2009]. The first method, keyboard input, is too slow for live captioning, and too ex-

pensive for transcribing shows in advance, and hence can be ruled out. The second approach is stenography, an abbreviated symbolic writing method that allows a trained person to type nearly as fast as a person can speak. It is still used commonly by broadcasting companies such as BBC, but the required three-year training makes it very expensive and calls for better methods. In fact, the third method, respeaking, has already become the most popular captioning method in the television industry. The required skills, ranging from knowledge about captioning and interpreting skills to a basic understanding of ASR technology, can be learned within weeks. Respeakers are trained to create television captions in real-time and often at a WER below 5% [Prazak et al., 2012]. To further reduce errors, script recognition [Evans, 2003] or post-correction [Homma et al., 2008] can be used. An initial ASR transcript can be used as a visual guide for the respeaker, and shot change detection can help improve the segmentation [Evans, 2003]. Unfortunately, ASR software is often only available as a black box, without optimization for respeaking. Moreover, live captioning through respeaking usually results in a summary rather than a faithful transcript. Recently, a combination of respeaking and typing was introduced for correcting an automatic transcript in real-time by one person [Prazak et al., 2012]. However, this method assumes a WER of only 10% for the initial ASR transcript, and requires a highly skilled respeaker and frequent breaks. In contrast, the proposed method requires no training, can be executed with fewer breaks, and allows the respeakers to stay closer to the original wording. In addition, none of the television-specific materials need to be relied on. However, these advantages are achieved by compromising some transcription speed.

2.5 Error Correction for Speech Input

Advances in speech recognition technology have enabled speech to become an important alternative to traditional input modalities that rely on a keyboard, mouse, or touch screen. However, speech can only be used effectively as an input method in combination with good error recovery mechanisms. While the corresponding research focuses on processing short, individual utterances presented by a single speaker, some of the methods found in literature can be transferred to our investigated topic of respeaking as a large-scale transcription method.

[Suhm and Waibel, 1997, Suhm et al., 2001] make a case for using multiple, orthogonal modalities for the correction of speech transcripts, such as spelling and handwriting recognition[1]. They further propose using available

[1] See also patent [Waibel et al., 1998]

context information to adapt the language model, and employ a rescoring strategy[1]. Results include the observation that depending on the typing speed and respeaking recognition accuracy, respeaking may be faster than typing. The experiment conditions are slightly different in that perfect accuracy desired and enabled by a post-correction step, but nevertheless the results are consistent with those reported in this thesis. Note that switching ASR systems, as proposed in [Zweig, 2009], can also be understood as such an orthogonal modality, the same holds for switching speakers as considered in this study. In [Vertanen and Kristensson, 2009], the automatic alignment of respoken corrections is investigated, which can be used in combination with automatic intention detection [Choi et al., 2012] to create a natural correction interface. The former method could also be used in the scenario investigated here, and enable respeaking segments partially.

Further research investigates the benefit of confusion networks [Mangu et al., 2000] for system combination. [Ogata and Goto, 2005] propose using confusion networks to obtain and display alternative word hypotheses for selection, and [Ishimaru et al., 2011] show that correction speed can be increased with richer confusion networks obtained by combining the output of several ASR systems. [Cesari et al., 2008] combine confusion networks from respoken corrections, and employ a "forced correction" strategy. [Vertanen and Kristensson, 2010] propose an improved method of combining confusion networks in the context of respeaking. In contrast, in this study we use a simpler approach of hypothesis combination based one-best hypotheses, because these turn out to enable more robust alignments in the context of combining utterances by multiple speakers, at least when enriched with phonetic information.

[1] See also patent [Waibel and McNair, 1998]

Chapter 3

The Proposed Method

3.1 Overview

The goal of this study is to develop a method to improve the quality of a speech transcript efficiently through the use of respeaking. We define efficiency as the word error rate reduction achieved in a certain amount of supervision time. Our approach comprises a sequence of steps, summarized as follows. As a preparative step, the respeaker undergoes an enrollment procedure. For a given speech that should be transcribed: (1) An initial automatic transcript is created. (2) Based on this initial transcript, the speech is segmented into short, sentence-like units. (3) Each of these segments is assigned a segment confidence score. (4) The respeaker speaks each segment. (5) The recognition hypotheses from the original speaker and the respeaker are combined to improve the results.

3.2 Preparative Step – Enrollment

We assume that respeaking is to be performed by the same, known speaker(s) repeatedly, which justifies the effort to train speaker-adapted acoustic models to enroll each speaker. The speaker records training material for supervised model adaptation, preferably in the same recording environment in which the respeaking is to take place. Depending on the amount of data, different adaptation techniques are suitable. In this study, we use maximum likelihood linear regression (MLLR) in a supervised fashion [Leggetter and Woodland, 1995, Gales, 1998].

MLLR is a method for acoustic model adaptation that attempts to find linear transformations for the mean values and covariance matrices of a Gaussian mixture HMM that can capture a general relationship between

the trained models and the particular speaker. More specifically, the transformations that yield maximum likelihood of the recorded data, given the transformed models, are estimated. A regression tree, determined by clustering the models in the acoustic space, represents groups of models that share the same transformation. The less adaptation data is available, the more shallow the tree, and the larger the number of models that share a transformation. MLLR can be applied in an unconstrained fashion [Leggetter and Woodland, 1995], with mean values and covariance matrices having different transformations. A second approach is constrained MLLR [Gales, 1998]: here, mean values and covariance matrices share the same transformation. We use both approaches simultaneously for an additive gain in accuracy.

3.3 Step 1 – Initial Recognition

As a visual guide for the respeaker, and to enable the succeeding steps, we use ASR to create an initial transcript from the original speech. We use confusion networks for decoding to estimate reliable confidence scores (see Section 3.5). The confusion networks are computed based on a semi-supervised segmentation (see Section 4.1). Besides the confidence scores, confusion networks might be helpful in the future for displaying word alternatives on the screen, as well as for improved hypothesis combination.

3.4 Step 2 – Segmentation

Next, the speech is segmented into smaller utterances. Segmentation is an important part of our approach, as it not only makes the actual respeaking easier, but also allows skipping segments via confidence filtering and simplifies navigation. Note that, as a limitation, segment-by-segment correction produces some overhead for each segment due to the delay that comes from the respeaker having to listen ahead before actually speaking. A suitable segmentation strategy should produce segments that are long enough to reduce this delay and ensure good recognition accuracy, but not so long that the respeaker has to speak more than is necessary. Segment breaks should also appear at natural positions in the sentence, as an awkward segmentation might be confusing and produce suboptimal language model scores when recognizing the respoken utterances.

3.4.1 Segmentation Algorithm

We adopt a method that attempts to obtain a natural segmentation and allows adjusting the segments' length [Matusov et al., 2006]. This segmentation scheme provides a good approximation of some of the desired properties stated above. The basic idea is to assign a probability to every possible segmentation, and then employ a dynamic programming strategy to find the globally optimal segmentation.

The probability of a segmentation is modeled as a log-linear feature combination:

$$P(s_0^m|w_1^n,t_0^n) = \frac{\exp(\sum_{l\in L}\lambda_l \cdot f_l(s_0^m,w_1^n,t_0^n))}{\sum_{m',s'^{m'}_0}\exp(\sum_{l\in L}\lambda_l \cdot f_l(s'^{m'}_0,w_1^n,t_0^n))} \tag{3.1}$$

Here, s_0^m are the indices of a segmentation into m segments, where the i-th segment corresponds to the word sequence $w_{s_i}^{s_{i+1}-1}$ (hence, $s_0 = 1$, and $s_m = n+1$). Further, w_1^n is the sequence of words to be segmented, consisting of n words, and t_i is the length of the prosodic break between w_i and w_{i+1}. λ_l is the feature weight and f_l the corresponding feature scoring function for all feature indices $l \in L$. The algorithm then finds the most probable among all combinations of segment breaks:

$$\text{argmax}_{m,s_0^m}\sum_{l\in L}\lambda_l \cdot f_l(s_0^m,w_1^n,t_0^n) \tag{3.2}$$

Here, we left out the denominator (since it does not depend on the choice of s_0^m), and the strictly increasing exponential function, without changing the result. The feature functions f_l assume the different segments to be stochastically independent and score each segment individually, yielding

$$f_l(s_0^m,w_1^n,t_0^n) = \prod_{i=0}^{m-1} f_l(w_{s_i}^{s_{i+1}-1},t_{s_i-1}^{s_{i+1}}) \tag{3.3}$$

We chose the following features:

- Segment-boundary language model score:

$$\begin{aligned} f_{\text{LM-bound}}(w_{s_i}\dots w_{s_{i+1}}) = & P_{\text{LM}}(<\text{s}>\ w_{s_i}\dots w_{s_i+K-1}) \times \\ & P_{\text{LM}}(w_{s_{i+1}-K+2}\dots w_{s_{i+1}}\ <\!/\text{s}>) \end{aligned} \tag{3.4}$$

  for an n-gram language model of order K.

- Inner segment language model score:

$$f_{\text{LM-inner}}(w_{s_i} \dots w_{s_{i+1}}) = P_{\text{LM}}(w_{s_i} \dots w_{s_{i-1}}) \tag{3.5}$$

- Length of the prosodic breaks before and after the segment, with a cutoff after 10 seconds, and normalized to values between 0 and 1:

$$f_{\text{pros}}(t_{s_i-1}, t_{s_{i+1}}) = 0.01 \cdot \max(t_{s_i-1}, 10\text{sec}) \cdot \max(t_{s_{i+1}}, 10\text{sec}) \tag{3.6}$$

- Segment length model, estimated on pre-segmented transcripts:

$$f_{\text{length}}(s_i, s_{i+1}) = P(\text{segment length } = s_{i+1} - s_i) \tag{3.7}$$

- Segment duration model, estimated on pre-segmented transcripts using a histogram approximation:

$$f_{\text{dur}}(s_i, s_{i+1}) = P(\text{segment duration } \approx t_{\text{end}}(s_{i+1}) - t_{\text{start}}(s_i)) \tag{3.8}$$

  where $t_{\text{start}}(w)$ and $t_{\text{end}}(w)$ denote the starting and ending times of word w.

- A penalty for inserting new segments, which can be used as a bias to increase or reduce the length of the segments, depending on its feature weight:

$$f_{\text{pen}}(s_i, s_{i+1}) = 1 \tag{3.9}$$

Our selection of features is similar but not equal to the features proposed by [Matusov et al., 2006]. More specifically, we consider boundary and inner language model probabilities as separate features and let the tuning procedure determine the relative importance of both. Also, we add the duration model to ensure the segments to be coherent units not only grammatically but also in a temporal sense. In contrast, the original method uses only a single language model score, and no duration model.

The search is performed using a dynamic programming approach. We assume that at a given time, we have already found the optimal segmentations for all word sequences (w_1), $(w_1 w_2)$, up unto $(w_1 \dots w_i)$, given any number of segments k up to a maximum of K. Let $s(j, k)$ denote the score corresponding to the best segmentation up to word w_j when dividing into k segments $(j \in \{1, \dots, i\}, k \in \{1, \dots, \min(K, j)\})$. Further, we keep track of the actual segmentation using $pred(j, k)$. This definition can be thought of as defining segment breaks before words $w_1, \dots, pred(pred(w_i, k), k-1), pred(w_i, k), w_i$, given a word sequence $w_1 \dots w_i$.

Now, adding the next word w_{i+1}, the best segmentation for the sequence $w_0 \dots w_{i+1}$ can be computed for all $k \in \{1, \dots, \min(K, i+1)\}$ by setting:

$$s(i+1,k) := \max_{j \in \{1,\dots,i\}} \left\{ s(j,k-1) + \sum_{l=1}^{L} \lambda_l \cdot f_l(w_{s_j}^{s_{i+1}}, t_{s_j-1}^{s_{i+1}}) \right\} \tag{3.10}$$

$$pred(i+1,k) := \operatorname*{argmax}_{j \in \{1,\dots,i\}} \left\{ s(j,k-1) + \sum_{l=1}^{L} \lambda_l \cdot f_l(w_{s_j}^{s_{i+1}}, t_{s_j-1}^{s_{i+1}}) \right\} \tag{3.11}$$

Here, $f_1, \dots, f_L$ denote our feature functions is stated above. In practice, we do not want segments to become arbitrarily long. We can thus define a maximum segment length L and let j run only in a smaller interval $j \in \{(i+1) - L, \dots, i\}$. In our experiments, we set L to 15. As a side effect, constraining the segment length makes the search run in linear time with respect to the number of words. More precisely, the segmentation algorithm now runs in $\mathcal{O}(N \cdot K)$ instead of $\mathcal{O}(N^2 \cdot K)$ for N words and K segments. The number of segments is approximately proportional to the number of words, hence we can finally approximate the complexity as $\mathcal{O}(N^2)$.

3.4.2 Tuning

We tuned the segmentation parameters on transcriptions for TED talks. These transcriptions are segmented as subtitles to accompany the videos. We found these subtitles to have a pleasant length for respeaking, although we did not perform an explicit optimization of the segment length for our task. Moreover, the segmentation is usually chosen in a way that the resulting segments form units that sound natural and often occur when the speaker pauses. Since only rough segment-level alignments are available, we used ASR to create accurate word-level forced alignments, which are needed to determine the prosodic feature for the segmentation. For the language model, we used a 4-gram model interpolated from TED data and other background corpora.

Powell's method was employed for the tuning, which performs a grid search for one parameter at a time, until a convergence criterion is reached. We used random initial parameters, and iterated through the parameters in random order. Since scaling the parameters uniformly does not change the result, we constrained the parameters so that the sum of their absolute values must equal 1.0. We considered the tuning as having converged when there was no parameter adjustment possible that improved the objective function by more than 0.00001.

Objective	$f_{\text{LM-bound}}$	$f_{\text{LM-inner}}$	f_{pros}	f_{length}	f_{dur}	f_{pen}
F-measure	0.027	0.111	0.109	-0.027	0.092	-0.635
Mean deviation	0.030	0.089	-0.0776	0.080	0.140	-0.583

Table 3.1: Sample parameter configurations, tuned according to f-measure and mean deviation.

In [Matusov et al., 2006], it is suggested to use the balanced f-measure as the objective function for tuning, which is defined as the harmonic mean of recall and precision (see [Rijsbergen, 1979]) and intuitively measures how many of the predicted segment breaks are correct, and how many of the actual segment breaks were predicted. The f-measure penalizes segment breaks that are just slightly off and those that are in the middle of a segment equally, which may not be optimal. Hence, we compared it to a different objective function, the *Segment Overlap Score (SOV)* [Zemla et al., 1999], which is used in bio-informatics as a measure of the degree to which the segments of a true and a hypothesized segmentation overlap. Unfortunately, the SOV seems to be to specific or "strict", because the tuning converged already after a few iterations without any substantial improvements. For this reason, we implemented a third and simpler objective, measuring the mean deviation of reference segments r_0^m from the hypothesized segments s_0^n as follows:

$$\left(1 - \frac{1}{m}\sum_{i=0}^{m} \min_{0 \le j \le n} \frac{|r_i - s_j|}{max\_length}\right) \cdot \left(1 - \frac{|m-n|}{\max(m,n)}\right)$$

This can be understood as the mean segment boundary deviation times the difference in number of segments, the latter making sure that the number of segments be similar to that of the reference segmentation. This third objective function resulted in good segmentations, and had better convergence behavior than the SOV, although tuning seemed more prone to ending in suboptimal local optima. Also, we noticed that the dependence on the prosodic feature was quite weak as compared to using the f-measure. Consider the example in Table 3.1, in which the prosodic feature weight even gets negative when tuning using the mean deviation, as opposed to a positive weight for the f-score. A low prosodic feature weight causes problems in practice, since segments with no leading and trailing prosodic breaks are harder to understand and respeak (see section 4.6.1). We hence decided to use the f-measure in our reference implementation. Tuning resulted in a final f-measure of 0.45.

3.4.3 On Re-Decoding

A natural successive step that could be examined for performance gains is re-decoding using the new segmentation. However, manual analysis showed that sometimes segment breaks occurred at inappropriate locations, due to recognition or alignment errors. Moreover, some segments are very short, so language model context would have to be carried over between segments for reliable recognition. These factors suggest that re-decoding would probably not bring further gains, so we did not attempt such experiments.

3.5 Step 3 – Segment Confidence Estimation

The next step, the estimation of confidence measures, is important because they allow us to identify segments with potentially high error rates. By first correcting these segments, either through respeaking or typing, we can reduce a larger number of errors in less time.

3.5.1 Word-level Confidence Measures

The perhaps most intuitive confidence measure for recognized words would be a direct computation of their posterior probability. Let $W = w_1 \dots w_n$ be a recognition hypothesis, X the acoustic data, and $\mathcal{R}_i^w$ the set of all reference strings that contain w at their i-th position, i.e. $\mathcal{R}_i^w = \{R = r_1 \dots r_m \in \mathcal{V}^m | m \geq i, r_i = w\}$ for vocabulary $\mathcal{V}$. The probability of finding w at position i is then given by

$$\begin{aligned} P(w_i = w|X) &= \sum_{R \in \mathcal{R}_i^w} P(R|X) \\ &= \sum_{R \in \mathcal{R}_i^w} \frac{P(X|R) \cdot P(R)}{P(X)} \\ &= \sum_{R \in \mathcal{R}_i^w} \frac{P(X|R) \cdot P(R)}{\sum_{R'} P(X|R') \cdot P(R')} \end{aligned} \tag{3.12}$$

Unfortunately, a direct, exact computation of the posterior probability is computationally intractable. In the ASR literature, there exist two main approaches to approximate confidence measures for words: one is based on feature combination, and one estimates posterior word probabilities from lattices. For some time, feature combination was the predominate strategy to compute confidence measures. However, as demonstrated by [Schaaf and

Kemp, 1997b] and refined by [Wessel et al., 2001], accurate posterior probabilities can be estimated from lattices using the forward-backward algorithm on the decoding lattice, while averaging over the different alignments for each word. The obtained posterior is only an approximation because of the search constraints and pruning when building the lattice, but is demonstrated to outperform methods based on feature combination. Since the authors do not report on the effect of the search beam size on the reliability of the posterior probabilities, we performed some experiments (see section 4.4.1).

An alternative method that has been shown to accurately estimate posterior word probabilities is based on confusion networks [Mangu et al., 2000]. Confusion networks allow the decoding of hypotheses that minimize the WER directly, as opposed to minimizing the sentence error rate as in traditional lattice-based decoding approaches. To this end, a complete alignment is approximated by merging links in the lattice until a total ordering is found. The best hypothesis is then determined by simply choosing the words with highest link probabilities at each point. These posterior probabilities can also be used as confidence measures.

For the sake of completeness, we have also implemented a confidence score using a support vector machine (SVM) to combine several features. We chose several of the features reported to be most helpful in [Schaaf and Kemp, 1997a], including word duration, number of phones, average duration per phone, duration of the longest and shortest phone, the number of language model back-offs necessary, and the number of active states for the last frame, averaged over three adjacent frames. Results are reported in section 4.4.1.

3.5.2 Segment-level Confidence Measures

While it is straight-forward to compute sentence-level posterior probabilities, perhaps normalized by the number of frames, this would lead us to optimize the sentence error rate. Previous work has shown that for decoding, more directly optimizing the word error rate, rather than the sentence error rate, yields slightly improved results, even though both are well correlated [Mangu et al., 2000]. Doing so requires considerable additional effort, but there is a second reason why we prefer going the extra mile: A confidence measure that is more directly related to the word error rate is very helpful to intuitively fix a particular threshold, which is necessary to apply confidence filtering in practice.

Given word posterior probabilities, we define the segment confidence score as their arithmetic mean. This is justified as follows. If our goal is to decrease the WER effectively, we should start by correcting the segments whose WER

is comparably high. The mean posterior approximates said WER:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} p(w_i|X) &= \frac{1}{n}\mathbb{E}[\mathrm{I}+\mathrm{S}] = \frac{1}{C+S+I}\mathbb{E}[I+S] \\ &\approx \frac{1}{C+S+D}\mathbb{E}[I+S+D] = \mathbb{E}[\mathrm{WER}(w_1^n)]\end{aligned} \tag{3.13}$$

Here, n denotes the recognized number of words, and C, S, I, and D denote the number of words corrected, substituted, inserted, or deleted from the reference. The approximation becomes an equality if we assume all errors to be substitution errors, i.e. $D = I = 0$.

3.6 Step 4 – Respeaking

3.6.1 Supervision Strategies

For respeaking, we define two supervision strategies. The first, more traditional strategy is *sequential* correction: Segments are corrected in temporal order, and every segment is presented to the respeaker regardless of its confidence. The second, proposed strategy is to make use of segment *confidences*: Segments are corrected in ascending order of confidence, and supervision can be aborted once a certain threshold is reached. The first strategy makes it easier for the respeaker to keep track of the speech's context, whereas the second strategy has the advantage of saving effort via the confidence filtering. Note that it would be easy and also reasonable to mix both strategies, i.e. using a sequential order but with confidence filtering; however, this would complicate interpretability of our results and is thus left for future work.

In practice, a respeaker would start listening to a segment, and start speaking while still listening. If the speaker notices that the original transcript is already correct, he would abort the recording and directly proceed to the next segment. This strategy of *skipping* segments that are already correct is effective both in saving time and increasing accuracy. The correction effort for skipped segments can be expected to be roughly equal to their playback duration, while all other segments take longer, due to the inevitable delay between listening and speaking.

3.6.2 Graphical User Interface

A good design of the respeaking user interface is of critical importance and has a strong impact on the respeaking efficiency. This chapter describes the user interface that was developed in parallel to the respeaking experiments

conducted in this study, and reflects the experiences we made in the process. A screen shot of the result can be seen in Figure 3.1. The user interface displays the automatically segmented initial transcript, aligned to a vertical waveform representing the speech. Segment confidences are visualized by coloring the segment's edge with a color between red (low confidence) and grey (high confidence). Manually verified segments are assigned a green color. The tool allows playback of the complete speech, or individual segments. For each segment, a respoken version can be recorded, either simultaneously to listening, or afterwards. As a backup input method, for example in the case of out-of-vocabulary (OOV) words, corrections may also be typed, with the tab key allowing fast selection and jumping between words. As indicated earlier, segments may be manually verified, meaning that the initial hypothesis is already correct and no correction is required. Doing so will automatically cancel respeaking recording, and jump to the next segment.

There are several ways to navigate through speech. One way is to use the scrollbar and select arbitrary segments as desired. Alternatively, the previous/next buttons can be used to navigate through the segments in temporal order, or in the order of lowest confidence. A confidence threshold can be set, which will mark segments having a confidence higher than the threshold as inactive, and cause these segments to be skipped when navigating between segments.

The interface is designed to support the two supervision strategies described in Section 3.6.1.

- The speech can be supervised via manual verification, meaning that the respeaker would listen to all segments in temporal order, verifying segments if they are already correct, or respeaking them otherwise. This is a reasonable setup if the whole speech is to be corrected.

- The speech can be corrected by relying on confidences. Either a threshold is set in order to be able to easily skip the correction of segments that are likely to be correct in the first place, or segments can be navigated in the order of lowest confidence, e.g. until a certain proportion is corrected or a time limit is reached. This setup is reasonable if the respeaker wants to save effort by correcting only part of the speech.

Since the enrollment procedure is an integral part of our approach, the user interface also provides a simple dialog to conduct enrollment recordings.

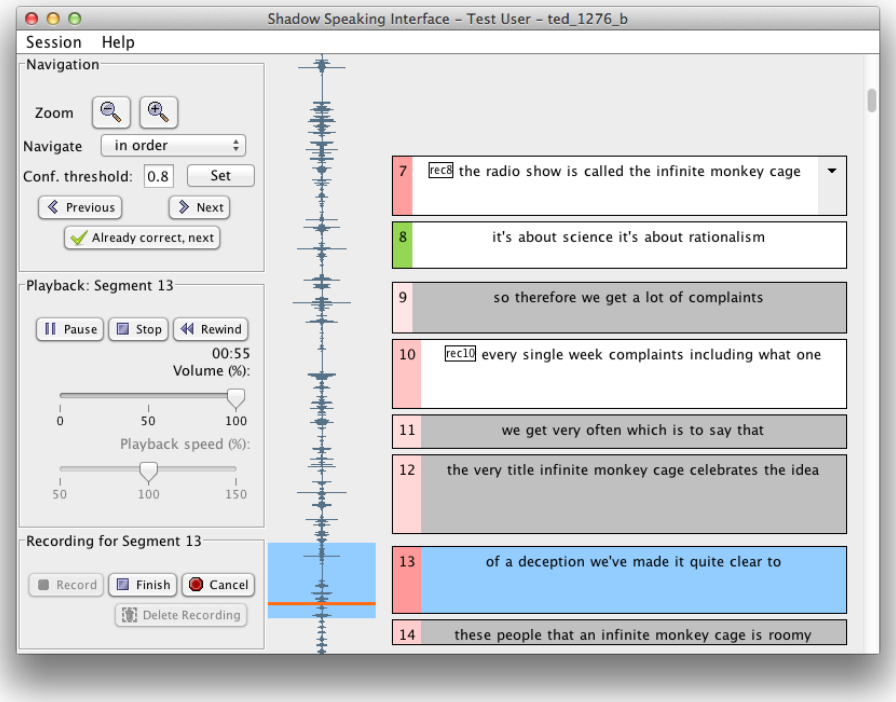


Figure 3.1: Screen shot of the respeaking interface.

3.7 Step 5 – Hypothesis Combination

An early error analysis of our respeaking experiments (see Chapter 4) revealed that our respeakers were able to correct about 60% of the original speakers' errors, but introduced 31% new errors. This surprising difference in occurring errors makes a strong case for using system combination techniques to combine both hypotheses, and hopefully have some errors cancel each other out. Unfortunately, traditional system combination methods for ASR rely on consistent time alignments between the hypotheses in order to create a global alignment. The time alignments in the respeaking scenario are not consistent, since recognition hypotheses from two different recordings are to be used. We hence use ROVER [Fiscus, 1997], a method for combining one-best hypotheses that works even without consistent time alignments. Two hypotheses are combined based on their word alignment, and the word with the highest confidence is chosen at each position. Null-links are assigned a fixed confidence score determined on a development set.

We noticed that in some cases, ROVER produced unstable alignments.

Consider the following hypothetical two examples of aligning two hypotheses:

(Hypothesis 1) I	have being	to Paris	
(Hypothesis 2) I	have been	Paris	

(Hypothesis 1) I	have being	to Paris	
(Hypothesis 2) I	have	been Paris	

Clearly, the first alignment of hypothesis 2 is preferable, but for ROVER both alignments are equally likely. To solve such problems, we extend ROVER by establishing word alignments based on the orthographic or phonetic similarity of words, rather than word identity. Still, in some cases even the orthographic or phonetic similarity is not useful to resolve ambiguities. Consider these example alignments:

(Hypothesis 1) I	do want	to eat	Sushi
(Hypothesis 2) I	like	to eat	Sushi

(Hypothesis 1) I	do want	to eat	Sushi
(Hypothesis 2) I	like	to eat	Sushi

The correct recognition in this example would be "I do like to eat Sushi." A possible scenario leading to the given two hypotheses might be an overly strong language model score leading to the substitution of the word "like" by "want" in the first hypothesis, and a deletion error for the word "do" in the second hypothesis. Often, these deletion errors occur when the ASR recognizes a filler instead of a lexical word. If we include the possibly occurring filler before the word "like" as a null-link, we can obtain the desired alignment.

Positive results for both extensions are reported in Section 4.5.

Chapter 4

Evaluation

We conducted detailed experiments comparing the effect of different correction strategies, speaking versus typing, and other factors that influence correction efficiency. We present our analysis in this chapter.

4.1 Setup & Data

For the evaluation of our method, we used data from TED[1], a platform for talks in the fields of technology, entertainment, and design for which recordings and transcripts are freely available. The talks are moderate in length (about 5 to 20 minutes), and are presented by skillful speakers producing fairly clear speech, making TED a good choice for performing respeaking experiments. We divided the TED corpus into training, development, and evaluation sets (see Table 4.1). The development set contained 3 complete talks with a total of 47 minutes and 6424 words. The evaluation data consisted of two 15-minute talks that were corrected fully and sequentially, and 5 talks that were corrected only partially (between 2 and 3 minutes per talk) and in order of segment confidence. Respeaking data was collected by 3 respeakers (1 native English, 2 foreign) for the development data, and 2 respeakers (1 native English, 1 foreign) for the evaluation data. The respeakers did not undergo any training procedure. All segments were both respoken and typed, in alternating order to remove bias. For evaluation, we measured the time spent for respeaking or typing for every segment. In addition, every respeaker recorded a text of 7416 words as enrollment material. All respeakers could be categorized as average speakers and above-average typists in terms of speed. For instance, their averaged typing and respeaking speeds

[1] www.ted.com

were 66 and 101 wpm, respectively, as compared to 46 and 107 wpm for an average typist and speaker according to [Moore et al., 2004].

We used a fairly standard decoding setup for our experiments. For acoustic modeling, MFCC with 3000 codebooks, 64 Gaussians, and a 42-dimensional feature vector were used. We used a 4-gram language model tuned to minimize the perplexity on a held-out TED data set. The vocabulary size was 180k. Decoding was performed by the IBIS decoder [Soltau et al., 2001] which is part of the *Janus Recognition Toolkit (JRTk)*. It used a semi-supervised segmentation of the talks based on a forced alignment of provided TED subtitles, where segment were divided when longer pauses were detected between subtitles. Training data for acoustic and language models includes TED, lecture material, TCSTAR[1], among others. On our development and evaluation sets, this setup achieved a word error rate of 27.1% and 21.7%, respectively. Note the superior accuracy for the evaluation set, which is partly due to more accurate transcripts as compared to the development set. For the development data, we used the closed captions provided by TED, which are optimized for readability, not perfect accuracy. For the evaluation data, on the other hand, manually corrected and improved transcripts were available. Also, better speakers for the evaluation dataset, and missing word normalization for the development transcripts might have impacted the word error rate.

Data set	Segment order	Respeakers	Talks	Words
DEV	sequential	3	3 (full)	6424
EVAL	sequential	2	2 (full)	2994
	by confidence		5 (partial)	2669

Table 4.1: Overview of respeaking data.

4.2 Speaker Adaptation

Table 4.2 shows results of the supervised speaker adaptation for the respoken development and evaluation sets, in comparison for our three respeakers. It can be seen that the enrollment was quite effective, decreasing the word error rate by roughly 25% relative on average. It is also interesting to see that in our experiments, the speaker adaptation had an especially strong effect on the first foreign speaker who had the worst baseline WER.

[1] www.tcstar.org

	DEV			EVAL	
	Native 1	Foreign 1	Foreign 2	Native 1	Foreign 1
baseline	21	25.3	20.9	17.2	22.4
speaker-adapted	17.3	16.5	17	14.5	15.3

Table 4.2: Word error rates before and after speaker adaptation.

4.3 Segmentation

We present a rather informal discussion on the usefulness of the proposed segmentation scheme in the respeaking context[1]. During the respeaking experiments, we found that our segmentation strategy produced a reasonable and effective segmentation into relatively natural sentence-like units. However, we noticed that sometimes segments were split at a position between two words where no pause was present. Even worse, sometimes segment breaks occurred in the middle of a word, due to erroneous automatic transcripts. This complicates understanding when listening to the segment, and creates ambiguity as to which segment the words lying on the segment boundaries belong to.

Table 4.3 displays the average time spend for correcting segments, as well as the resulting word error rates, in comparison between segment boundaries that obeyed or neglected prosodic pauses. It can be seen that for respeaking the time effort increased slightly, and for typing considerably, when no pause was present, possibly due to the respeaker having to replay the audio from before the segment break to understand the respective word. An interesting observation reported by the participants was that, in a way, respeaking was perceived as easier than typing. The reason is that for words that were hard to understand, one could still respeak just by mimicking the sound of it, whereas typing required knowing the correct spelling, which in turn required having completely understood the respective word. This is a possible explanation for the gap in time increase between typing and respeaking segments that did not obey prosodic breaks and thus often contained words at the segment boundaries that were hard to understand. In addition to supervision time, the resulting word error rate also increased, mostly because the respeaker either misunderstood the words near the segment boundary, or assigned them to the wrong segment.

To solve this problem, future segmentation methods could automatically detect prosodic breaks, and insert segment breaks only at these breaks. As

[1] A formal evaluation in which the user experiments are repeated on a number of different segmentations would be desirable, but too tedious to conduct.

	Avg. correction effort		Word error rate	
	Respeak	Keyboard	Respeak	Keyboard
Pause	6.6 sec	8.5 sec	18.2%	6.4%
No pause	6.8 sec	9.4 sec	20.2%	4.5%

Table 4.3: Difference of efficiency for segments having a pause before and afterwards, and segments with no pause either before or afterwards.

can be seen in the above numbers, we can expect significant efficiency gains when using such a segmentation scheme that considers prosodic breaks as a hard requirement.

The segment length was tuned towards the TED subtitles which are optimized for screen display, a reasonable but probably not optimal choice in terms of respeaking effort and speech recognition performance. Also, this tuning was performed on error-free reference transcripts, which are different from the actual automatic transcripts that we want to segment in practice. In future work, tuning should be carried out directly on automatic transcripts. This would also allow the inclusion of additional features, such as word confidences on the segment boundary. It should finally be investigated whether there are segmentation approaches that are more effective in a correction scenario, perhaps by explicitly optimizing the expected correction time.

Despite these issues, the method proved very effective for the respeaking scenario, as the results in the remainder of this chapter demonstrate.

4.4 Confidences

4.4.1 Evaluation: Word Confidence

The evaluation of confidence measures is not straightforward, as many possible ways of comparison exist. One might assume a binary classifier that labels every word with confidence above a threshold as correct, and below as incorrect. For such a classifier, accuracy-precision-curves compare accuracy and precision for different thresholds. These curves reveal valuable information for a specific confidence measure, and allow the comparison of different measures given a fixed dataset, i.e. a fixed decoding setup. In an initial experiment, we compared the performance of a feature combination approach using an SVM, and the direct estimation of posterior probabilities. Our best SVM used a quadratic kernel and performed slightly worse than the posteriors according to the accuracy-precision curve, which is in line with results found in literature [Jiang, 2005]. For example, at 75% recall, the posteriors

Decoding method	*Standard*					*Confusion network*
Search beam size	0.7	0.9	1.0	1.1	1.3	1.0
MCC	0.29	0.29	0.422	0.42	0.421	**0.480**
eff-word$_{15\%}$	0.75	0.75	0.268	0.254	0.243	**0.231**
eff-word$_{10\%}$	1.0	1.0	0.524	0.526	0.535	**0.426**

Table 4.4: Performance of different approaches to determining confidence measures, for lattices with different beam sizes, relative to the default size chosen for the decoding as a good trade-off between recognition accuracy and runtime. The last row shows results using a confusion network. The methods are compared using the Matthews correlation coefficient (MCC), and effort measures for achieving a word error rate of 15% and 10% using word-based correction.

reached a precision of 75%, compared to only 72% precision in case of the SVM. Based on these results, we decided to focus our further efforts on the posterior probabilities.

We are further interested in finding out how the decoding setup itself impacts the quality of the confidence measures. Since accuracy-precision-curves are difficult to compare across data sets, a better evaluation method is needed in this scenario. A popular choice for evaluation is the confidence error rate, which denotes the proportion of falsely predicted labels for a threshold that is optimal in this sense. Unfortunately, the confidence error rate suffers from the same problems. In particular, it is highly correlated with the word error rate, and hence does not allow a fair comparison across datasets. A better choice is the *Matthews correlation coefficient* (MCC, see [Matthews, 1975]), which basically measures the correlation between actual and predicted labels, taking on values between -1 and 1.

We compared the MCC for different lattice sizes by altering the search beam size. Table 4.4 shows that decreasing the search beam size for the computation of the confidence measures resulted in much inferior performance, but increasing it did not change the performance significantly in terms of the MCC. Confusion networks, for which we only provide numbers with unaltered search beam, performed considerably better, achieving a MCC of 0.48. These results suggest that the beam size used for the decoding is also a good choice for computing confidence measures, but the lattice-base approach is outperformed by confusion networks in any case.

As a more direct measure for the usefulness of different confidence measures to save correction effort, we define an effort measure as denoting the proportion of the transcript one must correct to obtain a desired WER for the transcript when proceeding in ascending order of confidences.

Note that for a scenario in which individual words are corrected sepa-

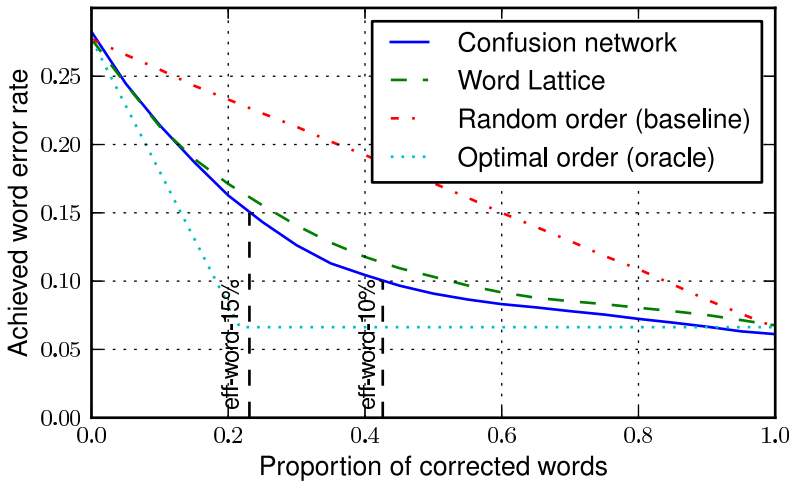


Figure 4.1: Word-based correction: Plot of the achievable word error rate, with respect to the proportion of corrected words. For comparison, the effort curves for an example correction in random order and for an optimal oracle correction order are shown.

rately, it is not clear how to cope with deletion errors. They may or may not overlap with (parts of) recognized words, and may possibly never be corrected at all. A possible solution would be to consider recognized fillers or pauses in between words as well, including the respective confidences. However, due to the missing language model context for fillers, their confidence scores might not be very reliable, and to keep matters simple, we assume in our simulation that deletion errors are never corrected in the word-correction scenario. Hence an error-free transcript cannot be achieved even when correcting all of the words in the hypothesized transcript.

Table 4.4 shows that computing confidence measures from the confusion network outperformed lattice-based methods in terms of correction effort also. It can be seen that increasing the lattice size reduced the correction effort only in some cases, while the MCC did not improve. The reduced effort might be owed to the slightly improved word error rate when using a bigger lattice to decode, rather than an improved accuracy of the confidence measure. Figure 4.1 shows the complete correction effort curves for lattice- and confusion-network-based confidence measures, in comparison to random and oracle correction order. Oracle in this context means that we assign every incorrect word a confidence of 0.0, and every correct word a confidence of 1.0. The figure reveals that part of the decrease in the effort for confusion

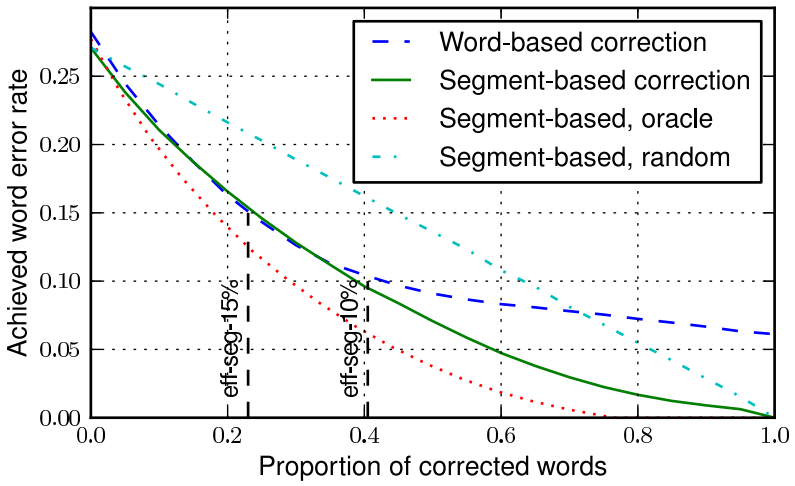


Figure 4.2: Segment-based correction: Plot of the achievable word error rate, with respect to the proportion of corrected words (the graph looked the same when plotting the proportion of corrected segments instead). For comparison, the effort curves for an example correction in random order, for an optimal oracle correction order, and for word-based correction are shown.

networks was due to the fact that these produced less deletion errors. Given a WER of about 28%, the use of this measure would make it necessary to correct 42.6% of all words to guarantee a WER of less than 10% (eff-word$_{10\%}$), and 23.1% of all words for a WER of less than 15%(eff-word$_{15\%}$). As can be seen in the figure, this lies roughly halfway between random and oracle correction orders. Additional experiments on subsets of the data confirmed the intuition that correction effort can be saved by lowering the word error rate of the initial transcript, the correlation being 0.89 for eff-word$_{10\%}$.

Since the confidence measure based on confusion networks performed best in terms of both classifier performance and resulting correction effort, we use it in our reference implementation.

4.4.2 Evaluation: Segment Confidence

To obtain a better understanding of how well our confidence measure performs in the respeaking context where whole segments instead of single words are corrected, we performed some further experiments to analyze our segment confidence measure. The segment error rate, denoting the proportion of segments containing at least one erroneous word, was 73.9% for the given baseline system and reference segmentation. This value can be understood

as the minimum number of segments that need to be corrected to achieve a perfect transcript, by assuming that it is known which segments are correct and which are not. By assigning segment confidences, and designing a threshold-based binary classifier as before, we obtained an MCC for segments of 0.466.

As a more direct measure, we computed the effort for segment-wise correction. A word error rate of less than 15% could be guaranteed by correcting at least 23% of all words (eff-seg$_{15\%}$), a word error rate of less than 10% by correcting at least 40.5% of all words (eff-seg$_{10\%}$), and a word error rate of less than 5% by correcting at least 60.6% of all words (eff-seg$_{5\%}$). Note that achieving a 5% WER has become possible using the new segment-based correction scheme, because deletion errors can be corrected now. Figure 4.2 shows that segment-based correction never takes more time than word-based correction, even though words with high confidence may be "corrected" sporadically as part of a segment low-confidence segment. When aiming for very accurate transcripts, segment-based correction outperforms word-based correction due to its ability to correct deletion errors. The figure further reveals the actual effort curve to be closer to the oracle curve, in which we assume the segments' confidence to be perfectly correlated to their WER, than to the curve for random correction order. As with words, the segment-based correction effort strongly depended on the WER of the data subset, with a correlation above 0.9.

4.4.3 Discussion

As a conclusion to the above experiments, the use of confidence measures can be expected to reduce the necessary correction effort strongly. Moreover, correction in a segment-based manner is beneficial in that it reduces correction effort and provides a means to correct deletions, besides being a necessity in respeaking. The required effort of a correction in the order of segment confidence was roughly halfway in between the baseline and the oracle scenario, leaving some room for improvement. There are three straightforward ways to reduce correction effort: (1) By reducing the word error rate of the initial transcript, a certain target word error rate is achieved faster. (2) Finding a segmentation scheme that more directly minimizes the segment error rate by clustering together correct or incorrect words, respectively, the segment-based oracle curve is moved closer to the word-based oracle curve. (3) More reliable confidence measures will help identify incorrect segments more reliably.

As a more distant future work, it would also be interesting to find a confidence measure that predicts the potential of the segment to be correct

Similarity measure	Word identity	Orthographic	Phonetic
Ignore fillers	16.7	16.4	16.2
Fillers as null-links	16.8	16.1	**15.9**

Table 4.5: Word error rates [%] for the hypothesis combination using different similarity measures and optionally including fillers as null-links. The baseline respeaking WER with no hypothesis combination was 16.9.

after respeaking. This confidence measure might for example give a low score to segments containing OOV words, since these will still be incorrect after respeaking. Also, we implicitly assumed that the correction effort for a segment is proportional to its length, and that ASR accuracy is independent of a segment's length, both of which are oversimplifications that bear further potential of improvement.

4.5 Hypothesis Combination

Table 4.5 shows the effect of different hypothesis combination strategies on the WER. The baseline word error rates were 27.1 (original speakers) and 16.9 (respeakers). It can be seen that orthographic and phonetic similarity measures by far outperform the strict word identity. Furthermore, the strategy of using recognized fillers as null-links improved the results for both orthographic and phonetic similarities.

4.6 Overall Performance

4.6.1 Word Error Rates

Table 4.6 shows resulting word error rates for our experiments. It can be seen that speaker enrollment is a crucial step of our method. Also, hypothesis combination yielded good results, although skipping over correct segments weakened the positive effect. The typing WER was 5.7%, which is perhaps surprising. Analysis showed that about 1.8% of that was due to segmentation issues, in which the lack of a prosodic break complicated understanding and caused ambiguity as to which segment a word belongs to. We conclude that a better segmentation strategy is crucial to improve the method. The remaining 3.9% WER were mostly due to ambiguous transcripts, e.g. caused by speaking mistakes of the original speaker.

	DEV	EVAL	
	all	all	skip
original speaker	31.4	21.7	-
baseline	22.4	19.8	15.8
speaker-adapted	16.9	14.9	12.3
hypothesis combination	**15.9**	13.1	**11.9**
keyboard	-	-	5.7

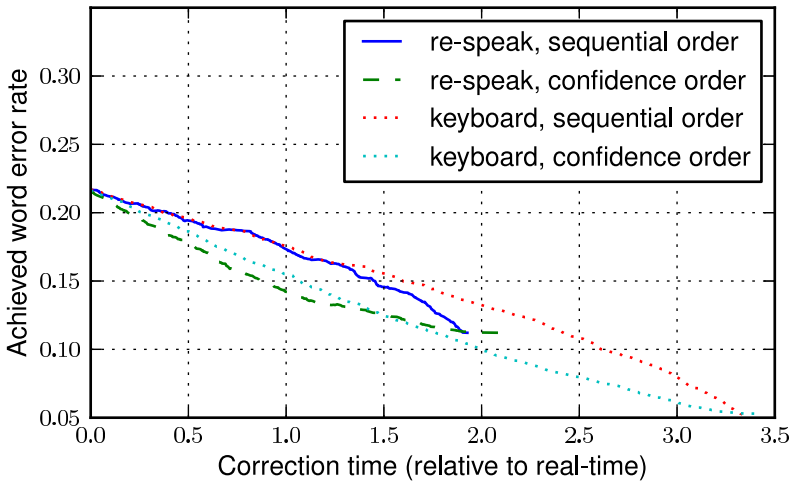
Table 4.6: Recognition word error rates [%] for baseline, speaker-adapted, and combined systems, and keyboard correction. Results differed when respeaking all segments, compared to skipping over segments that were already correct.

	Sequential	Confidence
Keyboard	61 wpm	58 wpm
Respeaking	**97 wpm**	83 wpm

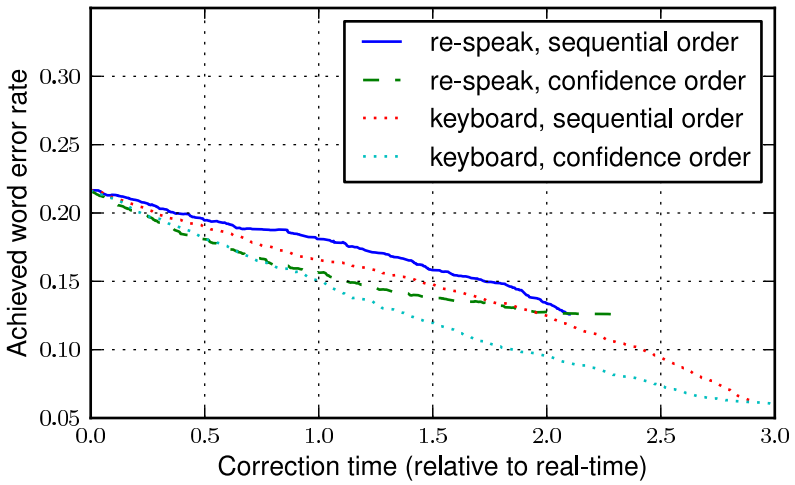
Table 4.7: Effective speaking and typing rates, including time needed to record or listen to a segment again, and saving time by skipping correction for segments that were already correct.

4.6.2 Correction Effort

Analysis of the correction time revealed a speaking rate of 189 wpm (words per minute) for the original speakers, and 131 wpm for the respeakers. Recall that the tested subjects were no professional respeakers, which may be the reason for the larger gap between original and re-speaking rates than reported in literature [Romero-Fresco, 2009]. The delay at the beginning of each recording, caused by having to listen ahead before respeaking, was 1.2 seconds on average and reduced the effective speaking rate to 100 wpm. There was significant additional overhead due to having to listen to a segment again when something was difficult to understand. Note that some of that overhead was due to segmentation issues and might thus be eliminated by a better segmentation. On the other hand, time was saved when the original transcript was already correct and the respeaking could be aborted early. Table 4.7 shows the speaking and typing rates when including all these factors. It can be seen that speaking was significantly faster than typing, though far from the original speaking rate. Also, for respeaking, proceeding in the order of lowest confidences decreased the speaking rate significantly, as the lack of context information made it harder to understand the speech.



(a) Efficiency curves for the native speaker.



(b) Efficiency curves for the foreign speaker.

Figure 4.3: Efficiency Curves.

4.6.3 Analysis of Efficiency

In this section, we analyze the efficiency of our approach, namely the achieved reduction in WER compared to the overall correction time spent. Figure 4.3 shows this relation over various scenarios. Figure 4.3(a) shows that for the native respeaker, respeaking or typing segments in the order of their confidence was more efficient than going in sequential order, unless all segments were to be corrected. In this case, confidences obviously do not bring any benefit, and the sequential order was more efficient due to the faster input rate as pointed out in the previous chapter. The diagram shows that the native speaker had better results with respeaking than typing when spending less than 1.5 and 2.5 times real-time for correcting in confidence- and sequential order, respectively. If one is willing to spend more time than that, lower word error rates can be achieved only through typing. In comparison to the native speaker, the foreign speaker's respeaking recognition accuracy was worse by about 12% relative, with the result that he was consistently more efficient by typing than respeaking (see Figure 4.3(b)). The scenarios of correcting in order of segment confidences versus sequential correction compare similar as for the native speaker.

4.6.4 Simulating a Better Speaker

In our experiments, the respeakers showed lower performance than reported in literature, both with respect to recognition rate and speaking rate. We thus performed some simulations that, even though they cannot capture the complex interplay of factors influencing efficiency in its entirety, demonstrate the further potential of the proposed method. We use slightly pessimistic values of 10% WER and 152 wpm speaking rate (80% of the original speaking rate), when compared to literature [Prazak et al., 2012,Romero-Fresco, 2009]. Figure 4.4 shows a simulation of a 10% WER speaker, which results in a noticeable efficiency gain. Next, we simulate doubling the segment length, and thus remove some of the overhead that is caused by the delay between listening and speaking. Perhaps surprisingly, this caused a loss of efficiency, as now the number of completely correct segments that can be skipped is much smaller. This indicates that the chosen segment length is already roughly a good value, despite not being explicitly optimized. Finally, we increase the speaking rate from 131 wpm to 152 wpm, which again results in a noticeable efficiency gain.

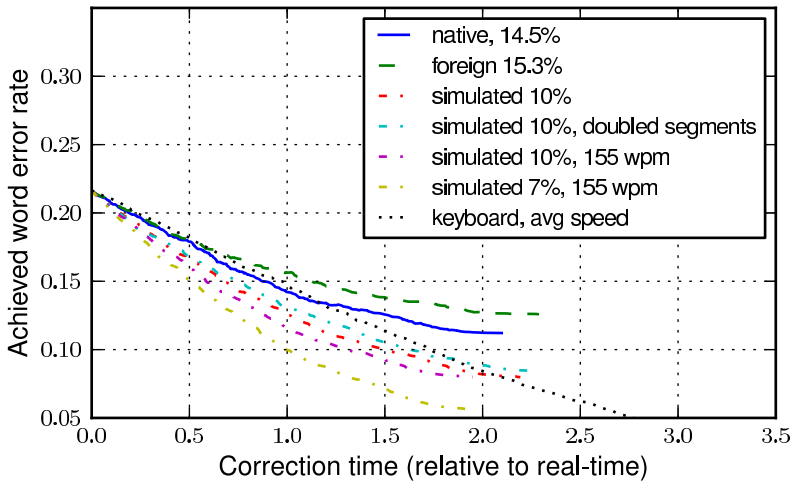


Figure 4.4: Efficiency curves for confidence-based correction by real and simulated speakers.

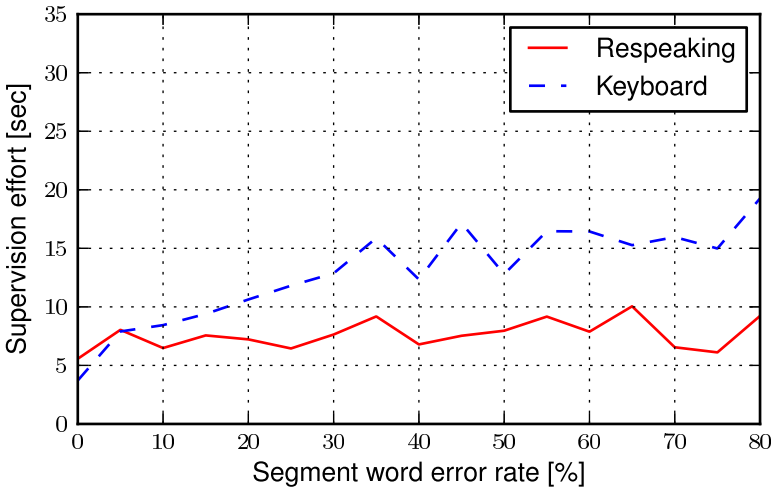
4.6.5 Respeaking Versus Typing

The previously described efficiency curves (Figures 4.3 and 4.4) demonstrate that depending on the targeted word error rate or the available time budget, either respeaking or keyboard-based correction should be chosen. However, we argue that the preferable correction method depends not only on the overall target, but also on each individual segment. For instance, our respeaking technique requires respeaking complete segments regardless of the number of errors in the original transcript. Typing, on the other hand, allows to correct only the incorrect words, which saves time especially when there are only few errors in the initial transcript. In fact, our respeakers naturally resorted to this strategy, as can be seen in Figure 4.5(a). We can see that respeaking effort remains roughly constant, whereas typing effort decreases as the segments' WER decreases. In particular, segments with a WER of 5% or less needed less time for typing than for respeaking, on average. This observation may be used to give suggestions to the user as to whether a segment should be typed or respoken, by estimating the WER based on the confidence score. Figure 4.5(b) shows that using our segment confidence measure, we can predict that segments with a segment confidence above 0.9 are faster typed than respoken.

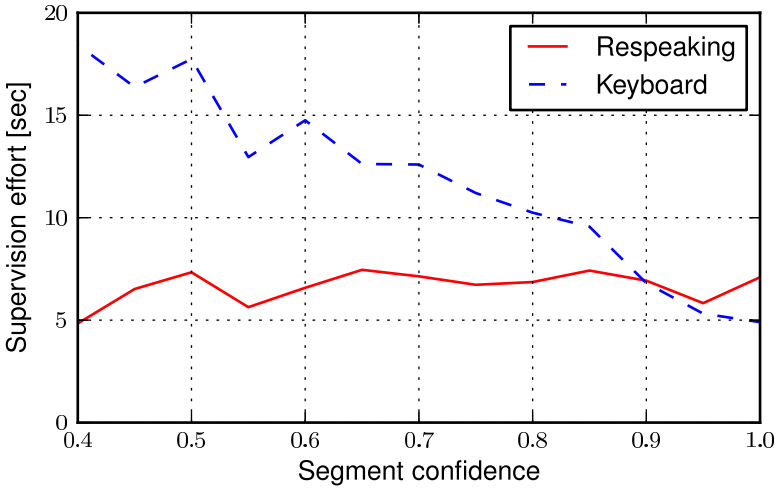
In general, the answer to the question which correction modality is more efficient, respeaking or typing, depends both on the time needed and the achieved reduction in WER. To predict the most efficient correction modal-

ity, both have to be modeled appropriately. Both the input rate and the accuracy should be modeled depending on the individual human corrector (see [Suhm et al., 2001] for initial work on this). The input rates also depend on the individual segment. For example the keyboard correction effort depends on the number of errors in a segment, as explained above. Similarly, WER reductions depend not only on the corrector but also on the segment. For example, the occurrence of OOV words puts respeaking at a disadvantage, since they cannot be recognized by common ASR technology. Another helpful clue to determine the expected respeaking recognition accuracy is the acoustic confusability.

An interesting observation we made in our experiments was the positive correlation between the segments' initial and respoken word error rate. This indicates that segments that were difficult for the recognizer originally were also more difficult to recognize when uttered by the respeaker. This can be explained by the fact that recognition errors are not only caused by difficult acoustic conditions, but also by difficult words and sentences. Consequently, the initial WER for a segment, as estimated by its confidence, can also be used as a measure of how difficult the recognition of a particular respoken utterance is.



(a) Segment WER versus correction effort.



(b) Segment confidence versus correction effort.

Figure 4.5: Comparison of keyboard and respeaking as input methods for individual segments.

Chapter 5

Conclusion

We proposed a method to enable efficient speech transcription through respeaking via a combination of various techniques. The proposed segmentation strategy succeeded in making the respeaker's task much easier. In our experiments, the respeakers were able to reduce the initial word error rate by 45% relative in about twice real-time. We showed that the efficiency strongly depends on the speaker's recognition rate, with respeaking outperforming typing for good speakers. We further demonstrated the potential of using segment confidences and hypothesis combination to increase efficiency, and showed that it depends on the particular segments whether respeaking or typing is a better choice.

Immediate improvement can be achieved by using a supervision strategy of proceeding sequentially, while using confidence filtering at the same time. An important point is the improvement of the segmentation. Hard requirement of a prosodic break, as well as an explicit optimization in terms of correction effort seem promising. Finally, results may be improved by using a more sophisticated hypothesis combination strategy, better ASR setup, and various adaptation strategies.

Chapter 6

Future Perspectives

In this final chapter, we would like to shed some light on the broader perspectives this work is intended to open up, if only as an initial step. We have previously contrasted the advantages and disadvantages of using automatic transcription as compared to human transcription. Specifically, automatic transcripts can be created fast and cheaply, but suffer from recognition errors. On the other hand, humans are more expensive and need breaks, but produce reliable results because they are strong at relating context information and performing error recovery. We are convinced that by investigating methods to bring ASR and human transcribers together, we can overcome some of the limitations of both. In a cooperative scenario, we would rely on the power of ASR to produce an initial guess and have a human corrector ensure transcription quality. Moreover, both would interact with and learn from one another. The presented respeaking approach allows transcribing more accurately than can be achieved by completely unsupervised methods, and faster than with using typing as input method. However, there is no real interaction or feedback, and no learning from the speech recognizer's part takes place.

In an ideal case, the computer should ask the human to correct only what is necessary, using the input method that is best in this particular case. Input methods may include typing, respeaking, or selecting from alternative words or word sequences. Uncertain parts should be verified by the user as efficiently as possible, for example by increasing playback speed or simply displaying (parts of) the transcript. Every user interaction should be designed not only to correct the respective errors efficiently, but also in a way that the ASR system can be improved most effectively. For instance, the user may be able to guess the reason that caused a particular error, and give feedback to adapt the ASR system appropriately. Every time the ASR system learns, a new improved hypothesis for the remainder of the transcript would

be created. Finally, user interactions should be managed in the particular order in which the ASR algorithms can learn most efficiently.

Bibliography

[Cesari et al., 2008] Cesari, F., Franco, H., Myers, G. K., and Bratt, H. (2008). MUESLI: Multiple Utterance Error Correction for a Spoken Language Interface. In *Interspeech*, pages 199–202, Brisbane, Australia.

[Choi et al., 2012] Choi, J., Kim, K., Lee, S., Kim, S., Lee, D., Lee, I., and Lee, G. G. (2012). Seamless Error Correction Interface For Voice Word Processor. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4973–4976, Kyoto, Japan.

[Evans, 2003] Evans, M. J. (2003). Speech Recognition in Assisted and Live Subtitling for Television. *BBC Research & Development White Paper*.

[Fiscus, 1997] Fiscus, J. G. (1997). A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER). In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 347–354, Santa Barbara, California, USA.

[Gales, 1998] Gales, M. J. F. (1998). Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. *Computer Speech and Language*, 12(2):75–98.

[Homma et al., 2008] Homma, S., Kobayashi, A., Oku, T., Sato, S., Imai, T., and Takagi, T. (2008). New Real-Time Closed-Captioning System for Japanese Broadcast News Programs. In *International Conference on Computers Helping People with Special Needs (ICCHP)*, pages 651–654, Linz, Austria.

[Hsu and Glass, 2009] Hsu, B.-J. P. and Glass, J. (2009). Language Model Parameter Estimation Using User Transcriptions. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4805–4808, Taipei, Taiwan.

[Huggins-Daines and Rudnicky, 2008] Huggins-Daines, D. and Rudnicky, A. I. (2008). Interactive ASR Error Correction for Touchscreen Devices.

In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL HLT)*, pages 17–19, Columbus, Ohio, USA.

[Ishimaru et al., 2011] Ishimaru, S., Nishizaki, H., and Sekiguchi, Y. (2011). Effect of Confusion Network Combination on Speech Recognition System for Editing. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Xi'an, China.

[Jiang, 2005] Jiang, H. (2005). Confidence measures for speech recognition: A survey. *Speech Communication*, 45(4):455–470.

[Kubat et al., 2007] Kubat, R., DeCamp, P., Roy, B., and Roy, D. (2007). TotalRecall : Visualization and Semi-Automatic Annotation of Very Large Audio-Visual Corpora Categories and Subject Descriptors. In *International Conference on Multimodal Interfaces (ICMI)*, pages 208–215, Nagoya, Japan.

[Leggetter and Woodland, 1995] Leggetter, C. J. and Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech & Language*, 9:171–185.

[Luz et al., 2008] Luz, S., Masoodian, M., and Rogers, B. (2008). Interactive visualisation techniques for dynamic speech transcription, correction and training. In *International Conference on Human-Computer Interaction Design Centered HCI (CHINZ)*, pages 9–16, Wellington, New Zealand.

[Mangu et al., 2000] Mangu, L., Brill, E., and Stolcke, A. (2000). Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

[Matthews, 1975] Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim Biophys Acta*, 405(2):442–451.

[Matusov et al., 2006] Matusov, E., Mauser, A., and Ney, H. (2006). Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 158–165, Kyoto, Japan.

[Moore et al., 2004] Moore, R. K., Court, R., and Street, P. (2004). Modelling Data Entry Rates for ASR and Alternative Input Methods. In *Interspeech*, Lisbon, Portugal.

[Ogata and Goto, 2005] Ogata, J. and Goto, M. (2005). Speech Repair: Quick Error Correction Just by Using Selection Operation for Speech Input Interfaces. In *Eurospeech*, pages 133–136, Lisbon, Portugal.

[Prazak et al., 2012] Prazak, A., Loose, Z., Trmal, J., Psutka, J. V., and Psutka, J. (2012). Novel Approach to Live Captioning Through Respeaking: Tailoring Speech Recognition to Re-speaker's Needs. In *Interspeech*, pages 4193–4196, Portland, Oregon, USA.

[Rijsbergen, 1979] Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, 2nd edition.

[Rodriguez et al., 2007] Rodriguez, L., Casacuberta, F., and Vidal, E. (2007). Computer Assisted Transcription of Speech. In *Pattern Recognition and Image Analysis*, pages 241–248.

[Romero-Fresco, 2009] Romero-Fresco, P. (2009). More haste less speed: Edited versus verbatim respoken subtitles. *Vigo International Journal of Applied Linguistics*, 6:109–134.

[Sanchez-Cortina et al., 2012] Sanchez-Cortina, I., Serrano, N., Sanchis, A., and Juan, A. (2012). A prototype for Interactive Speech Transcription Balancing Error and Supervision Effort. In *International Conference on Intelligent User Interfaces (IUI)*, pages 325–326, Lisbon, Portugal.

[Schaaf and Kemp, 1997a] Schaaf, T. and Kemp, T. (1997a). Confidence Measures for Spontaneous Speech Recognition. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 875–878, Honolulu, Hawaii, USA.

[Schaaf and Kemp, 1997b] Schaaf, T. and Kemp, T. (1997b). Estimating confidence using word lattices. In *EuroSpeech*, pages 3–6, Rhodes, Greece.

[Soltau et al., 2001] Soltau, H., Metze, F., Fügen, C., and Waibel, A. (2001). A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 214–217, Madonna di Campiglio, Italy.

[Suhm et al., 2001] Suhm, B., Myers, B., and Waibel, A. (2001). Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8(1):60–98.

[Suhm and Waibel, 1997] Suhm, B. and Waibel, A. (1997). Exploiting repair context in interactive error recovery. In *Eurospeech*, pages 1659–1662, Rhodes, Greece.

[Vertanen and Kristensson, 2009] Vertanen, K. and Kristensson, P. O. (2009). Automatic selection of recognition errors by respeaking the intended text. In *Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 130–135, Merano, Italy.

[Vertanen and Kristensson, 2010] Vertanen, K. and Kristensson, P. O. (2010). Getting it Right the Second Time: Recognition of Spoken Corrections. In *Workshop on Spoken Language Technology (SLT)*, pages 289–294, Berkeley, California, USA.

[Waibel and McNair, 1998] Waibel, A. and McNair, A. E. (1998). Locating and Correcting Erroneously Recognized Portions of Utterances by Rescoring Based on Two N-Best Lists.

[Waibel et al., 1998] Waibel, A., Suhm, B., and McNair, A. E. (1998). Method and Apparatus for Correcting and Repairing Machine-Transcribed Input Using Independent or Cross-Modal Secondary Input.

[Wessel et al., 2001] Wessel, F., Schlüter, R., Macherey, K., and Ney, H. (2001). Confidence Measures for Large Vocabulary Continuous Speech Recognition. *Speech and Audio Processing*, 9(3):288–298.

[Zemla et al., 1999] Zemla, A., Venclovas, C., Fidelis, K., and Rost, B. (1999). A Modified Definition of Sov, a Segment-Based Measure. *Proteins: Structure, Function, and Bioinformatics*, 34(2):220–223.

[Zweig, 2009] Zweig, G. (2009). New Methods for the Analysis of Repeated Utterances. In *Interspeech*, pages 2791–2794, Brighton, UK.