Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled

Marijn Huijbregts

CTIT Ph.D. thesis Series No. 08-123
Centre for Telematics and Information Technology (CTIT)
P.O. Box 217 - 7500 AE Enschede - The Netherlands

SIKS Dissertation Series No. 2008-26
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN 978-90-365-2712-5

ISSN 1381-36-17 (CTIT Ph.D. thesis Series No. 08-123)

Printed by PrintPartners Ipskamp, Enschede, The Netherlands

SEGMENTATION, DIARIZATION AND SPEECH TRANSCRIPTION: SURPRISE DATA UNRAVELED

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 21 november 2008 om 13.15 uur

door

Marijn Anthonius Henricus Huijbregts

geboren op 29 oktober 1976
te Renkum

Dit proefschrift is goedgekeurd door:

Prof. dr. F.M.G. de Jong (promotor)
dr. R.J.F. Ordelman (assistent-promotor)

ACKNOWLEDGEMENTS

I like programming. Put me in a room with a computer (or preferably more than one), provide me with a challenging task and I will probably enjoy myself solving puzzles until my four years of funding run out. At the Human Media Interaction (HMI) group of the Department of Electrical Engineering, Mathematics and Computer Science at the University of Twente, I *was* provided with a computer and a challenging task, but luckily also with the support of some great people that helped me to complete my research. I would like to thank everybody at HMI who directly or indirectly supported me these four years.

In particular I would like to thank my daily supervisor Roeland with whom I have enjoyed countless discussions before, during and after our long lunch walks. My supervisor Franciska also was of great help, especially during the writing process of this thesis. I'm thankful that both Roeland and Franciska allowed me to explore many research directions but also made sure that I wasn't overdoing it.

A lot of people outside HMI helped as well. It was a lot of fun discussing various ASR related topics with David. He pointed out interesting issues more than once and he convinced me to go abroad for an internship at the International Computer Science Institute (ICSI).

At ICSI I worked with Chuck on speaker diarization. I have never met anybody as enthusiastic as Chuck, both in work as in pronouncing my name. He always made time to discuss the results of my experiments and, above all, we've had a lot of fun.

For all the fun that I have had in Berkeley, I also want to thank everybody at ICSI and the 'Dutchies'. Because of the people at ICSI I felt at home in Berkeley instantly. Thanks for the great lunch talks, foosball matches, barbecues, sight-seeing, movies and of course the great nights at the pub. Thanks to the Dutchies for the same thing. I hope you've burned those Halloween pictures.

Back home there were also quite some people that distracted me from work. Thanks for this to all my friends and family, as to be honest, I don't think I actually would have enjoyed sitting behind a computer for four solitary years. Thanks to Piranha and my team mates for some great years of playing water polo. Thanks to the Gonnagles for some great years of playing tunes.

The biggest distraction from work in the last phase of my research has been Janneke. Spending time with her in front of Taj Mahal, on the beach at Ameland or just in our backyard made me realize that in fact Janneke is not distracting me from work, but work is distracting me from Janneke.

My parents have always supported me in everything I do, from long before I started my PhD until today, and for this I want to thank them most of all.

The work reported in this thesis was supported by the BSIK-program MultimediaN which is funded by the Dutch government under contract BSIK 03031. Part of my work was also funded by the European Union 6th FWP IST Integrated Project AMIDA (Augmented Multi-party Interaction with Distant Access, FP6-506811). I would like to thank these organizations for funding my research and providing a platform for interaction with other researchers. I would also like to thank all other projects and organizations with which I have worked. In appendix B you can find a short list of these projects.

Marijn

CONTENTS

1 Introduction **1**

1.1 Fading memories . . . 1
1.2 Locked archives . . . 2
1.3 Unlocking archives . . . 2
1.4 Information retrieval . . . 4
1.4.1 Image retrieval . . . 4
1.4.2 Video retrieval . . . 5
1.4.3 Spoken document retrieval . . . 5
1.5 Automatic speech recognition for SDR . . . 6
1.5.1 Keyword spotting and sub-word recognition . . . 6
1.5.2 Large vocabulary continuous speech recognition . . . 7
1.5.3 Sub-word recognition or LVCSR? . . . 8
1.5.4 Segmentation and clustering . . . 8
1.5.5 Computational constraints . . . 8
1.5.6 Statistical models . . . 9
1.5.7 LVCSR for broadcast news recordings . . . 10
1.5.8 Robustness . . . 10
1.5.9 Towards LVCSR for surprise data . . . 10
1.6 About this thesis . . . 12
1.6.1 Research goals . . . 12
1.6.2 Development requirements . . . 14
1.7 Thesis outline . . . 15

2 State-of-the-art in ASR **17**

2.1 Fundamentals . . . 17
2.1.1 Feature extraction . . . 18
2.1.2 Hidden Markov Models . . . 18
2.1.3 Gaussian Mixture Models . . . 19
2.1.4 Viterbi . . . 20
2.1.5 Acoustic models . . . 21

2.1.6 Language model and vocabulary . . . 22
2.1.7 Dictionary and pronunciation prefix tree . . . 23
2.1.8 The decoder . . . 23
2.1.9 Decoder assessment . . . 25
2.2 NIST benchmarks for ASR . . . 26
2.2.1 NIST Benchmark procedures . . . 27
2.2.2 Benchmark results . . . 27
2.2.3 The broadcast news benchmark series . . . 28
2.2.4 The 10xRT broadcast news benchmark series . . . 30
2.2.5 The meeting rich transcription benchmark series . . . 30
2.2.6 Existing techniques for creating robust systems . . . 32
2.3 Segmentation . . . 32
2.3.1 Feature extraction for segmentation . . . 33
2.3.2 Silence-based segmentation . . . 33
2.3.3 Model-based segmentation . . . 34
2.3.4 Metric-based segmentation . . . 34
2.3.5 Assessment of segmentation systems . . . 35
2.4 Clustering and speaker diarization . . . 36
2.4.1 Agglomerative clustering . . . 37
2.4.2 Assessment of speaker diarization systems . . . 38
2.4.3 NIST benchmark series for speaker diarization . . . 39
2.4.4 The ICSI speaker diarization system . . . 39
2.5 Techniques for robust ASR . . . 43
2.5.1 Feature and acoustic model normalization . . . 43
2.5.2 Acoustic model adaptation . . . 45
2.5.3 Speaker adaptive training . . . 47
2.6 Final remarks . . . 47

3 The SHoUT system **49**
3.1 Development strategy:
the fewer parameters the better . . . 49
3.2 Software architecture, some definitions . . . 50
3.3 System description . . . 51
3.3.1 Speech activity detection . . . 52
3.3.2 Segmentation and clustering . . . 53
3.3.3 Automatic speech recognition . . . 54
3.3.4 Acoustic model adaptation . . . 55
3.4 Summary . . . 55

4 Speech activity detection **57**
4.1 What is considered speech? . . . 58
4.2 The algorithm and its steps . . . 58
4.2.1 Bootstrapping . . . 60
4.2.2 Training the models for non-speech . . . 60
4.2.3 Training all models . . . 60

4.2.4 Training speech and silence models 61
4.3 Feature extraction 61
4.4 Confidence measures 62
4.5 The bootstrapping component:
Dutch broadcast news SAD 63
4.6 System parameters 64
4.7 Evaluation 65
4.7.1 Broadcast news evaluation 66
4.7.2 Out-of-domain evaluation 66
4.7.3 The IDIAP speech/music evaluation 66
4.7.4 Dutch TRECVID$_{07}$ ASR evaluation 67
4.8 SAD for speaker diarization 67
4.9 Conclusions and future work 68

5 Speaker diarization 71
5.1 Agglomerative model-based speaker diarization 72
5.2 The RT06s submission, SHoUT$_{D06}$ 73
5.2.1 System description 74
5.2.2 RT06s evaluation 75
5.2.3 Post evaluation changes 76
5.3 SHoUT$_{D06}$ system analysis 76
5.3.1 RT06s post-evaluation test set 77
5.3.2 Comparison to the ICSI RT06s system 78
5.3.3 Oracle experiments 81
5.3.4 Reference transcripts 81
5.3.5 Experimental set-up, six oracle experiments 82
5.3.6 Experiment results 84
5.3.7 Discussion and conclusions 85
5.4 The RT07s submission, SHoUT$_{D07}$ and ICSI-RT07s 85
5.4.1 The ICSI-RT07s speaker diarization system 86
5.4.2 Test set 86
5.4.3 Speech activity detection 87
5.4.4 Smoothing SAD 87
5.4.5 Blame assignment 88
5.4.6 Noise Filtering 89
5.4.7 Delay Features 90
5.4.8 Discussion 90
5.5 Speaker diarization for long recordings 91
5.5.1 The Cut&Mix system, SHoUT$_{DCM}$ 91
5.5.2 The multiple merges system, SHoUT$_{D07*}$ 95
5.6 Conclusions and future work 97
5.6.1 System analysis 97
5.6.2 Diarization for long recordings 98

6 Automatic speech recognition **99**
6.1 Modular decoder design . . . 100
6.1.1 Language models . . . 101
6.1.2 Acoustic models . . . 102
6.1.3 Pronunciation prefix tree . . . 104
6.1.4 Token-passing architecture . . . 105
6.2 Search space management . . . 106
6.2.1 Token pruning . . . 107
6.2.2 Language model look-ahead . . . 108
6.2.3 Experiments . . . 110
6.2.4 Discussion . . . 112
6.3 Robust ASR . . . 113
6.3.1 Vocal tract length normalization . . . 114
6.3.2 Acoustic model adaptation . . . 116
6.3.3 Evaluation of the robustness techniques . . . 117
6.3.4 Discussion . . . 120
6.4 Conclusions and future work . . . 120

7 System evaluation **123**
7.1 N-Best . . . 123
7.1.1 System description . . . 124
7.1.2 Evaluation results . . . 127
7.1.3 Post-evaluation analysis . . . 127
7.1.4 Conclusions and discussion . . . 130
7.2 Surprise data: TRECVID . . . 130
7.2.1 System description . . . 131
7.2.2 Evaluation results . . . 131
7.3 Conclusions . . . 132

8 Conclusions **135**
8.1 Research goals and future directions . . . 135
8.1.1 Segmentation . . . 135
8.1.2 Speaker diarization . . . 137
8.1.3 Automatic speech recognition . . . 138
8.1.4 The sum of the three subsystems:
the full SHoUT system . . . 139
8.2 Development goals . . . 139
8.3 Extending the horizon for SHoUT . . . 140
8.3.1 Automatic speech recognition . . . 140
8.3.2 Spoken document retrieval . . . 141

Appendices **144**

A Data collections **147**
A.1 Spoken Dutch Corpus 147
A.2 N-Best 147
A.3 Twente news corpus 148
A.4 TRECVID$_{07}$ data set 148
A.5 Rich Transcription benchmark for meetings 149
A.6 The IDIAP speech/music evaluation set 149

B Projects and demonstrators **151**
B.1 MultimediaN 151
B.2 AMI and AMIDA 152
B.3 CHoral 152
B.4 N-Best 152
B.5 TRECVID 153
B.6 MESH 153
B.7 MediaCampaign 153

CHAPTER 1

INTRODUCTION

1.1 Fading memories

Our holiday had been great. We drove through California in only two weeks, but that was enough to collect stories that will stay with us for years to come. And thanks to modern technology I have all those memories available on my laptop. It contains more than five hundred photos taken with my digital camera, an endless amount of video footage and even some video clips taken with the camera of my phone in some pub that I'd rather forget about altogether.

Tonight we're meeting to relive our adventure for the first time and of course, to exchange our best photos and videos. I did it again though: lost in my latest coding project, I forgot all about the time and now there is only little time left to prepare the dessert I'm supposed to bring with me. Although I'm sure that the recipe must be somewhere in my well organized recipe book, I decide to just search for it on the internet and save myself some time.

With the fruit cocktail under one arm and my laptop under the other I struggle to my car. I don't even have to search for my friends phone number, I just yell out his name to my phone and a few seconds later I'm explaining to him that I'll be a bit late because of this null pointer in my code.

I'm just finished explaining what a pointer is when my car navigation system tells me that I have reached my destination. The technology saved my day. That is, until we're ready to watch our holiday footage.

We only skimmed through a tenth of our digital video archive before we loose interest. We now realize that what we've had for breakfast at the third day of our vacation wasn't that exciting. Unfortunately, it's hard to find the nice bits in between all these hours of useless chatter. After a while I'm almost sorry that I've accidentally deleted those video clips from the pub. We decide to forget about it and go out to capture some new ones.

1.2 Locked archives

Nowadays, creating large digital multimedia archives is no problem. The story in the previous section contains a few examples of such archives. On the internet an unlimited amount of recipes is stored, car navigation systems contain detailed maps of entire continents and modern phones can store huge amounts of phone numbers. It is easy to retrieve information from these three archives because special measures have been taken to make the archives searchable. The holiday footage collection though did not contain any extra tooling that could aid in searching for interesting fragments. Due to the ever declining costs of recording audio and video (the footage would fit on a disk of only 200 euros), the data set was easily created in only two weeks, but because of its density, the information in the data set is hard to retrieve.

This problem is not limited to home made video archives. It is becoming more common for example to record lectures[1] or governmental or corporate meetings. Without special care, finding a specific lecture in an entire archive can be difficult. In a project called the ‘Multilingual Access to Large spoken ArCHives’ (MALACH) a huge number of Holocaust survivors was interviewed. This resulted in oral narratives of in total $116,000$ hours of video in 32 different languages [Oar04]. It is obvious that an archive this big is of little use without proper facilities to unlock its information. To the extreme, although it might seem a bit strange to do so, in [Chu03] it is shown that soon it will be affordable to store everything that a person sees, hears, says, reads and writes in his entire lifetime on a single disk. As long as it is not possible to search such a huge archive for interesting information, it is useless indeed to store a lifetime of data. But if search facilities would be available, suddenly such an archive might be very interesting to create. For some textual archives this is already the case. For example, most people do a similar thing with their email accounts. They don’t throw anything away, but instead just search their email archive when they need to.

1.3 Unlocking archives

Books often provide search facilities using an *index*. An index contains all important words with the pages they occur on in an alphabetically ordered list. Systems that provide automatic search in textual archives such as an email archive or search engines on the internet often work in a similar fashion. Each time a document is added to the archive, the index is updated with the words from the document. When the user enters a *query*, describing his information need, the words from the query are looked up in the index and a list with relevant documents is created.

Searching multimedia archives is less straightforward as searching textual archives. In *text retrieval*, a query is typically of the same modality as the matching word in the index. They both are represented by a sequence of characters and therefore it is easy to match them. In case of multimedia archives, often the modality of the query and the content do not match. The query is often formulated in written text or sometimes in speech as in the example of the mobile phone, whereas the content

[1] for example `http://videolectures.net`

of multimedia archives are moving images, sounds and speech. This phenomenon is called the *representation mismatch* [Ord03].

The representation mismatch can be solved by either translating the query into the format of the content, by translating the content into the format of the query, or by translating both into a convenient third format. Program guides make it possible to search for interesting programs in radio or television broadcasts by converting multimedia content into textual form. A car navigation system translates the query into coordinates (an address is queried in text, translated into coordinates and compared to the database). Before the telephone number of a friend can be retrieved by speech from a mobile phone, the user first needs to provide the phone with the pronunciation of his friends name, a so called voice label. Although it is hidden for the user, the mobile phone maps both the query (the pronunciation of the friends name when his phone number is needed) as the voice label into a third mathematical representation that allows for comparison of the two.

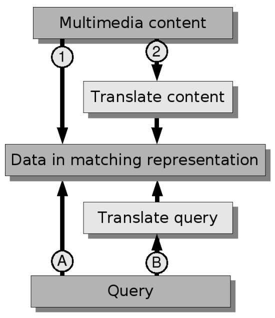


Figure 1.1: *Solving the representation mismatch between content and query in a multimedia retrieval system. The multimedia content is either indexed directly (1) or after it has been translated to an alternative representation (2). If the representation of the query matches the representation of the content, it can be used directly for retrieval (A). Otherwise it is translated before being used (B).*

In the program guide example, the name of each television program and the time that it will broadcast are manually transcribed so that it is possible to switch to a channel at the appropriate time instead of surfing aimlessly through all television channels. In this example it is possible to solve the representation mismatch manually, but in a lot of other cases, manually solving the representation mismatch is too time consuming and therefore too expensive. For example, although it would unlock the holiday video collection, it would be very time consuming to manually write down the details of each single fragment of all video footage. Even if this could be done while playing the raw footage, it would take as long as the footage itself. For the holiday videos this means it could be done in two weeks, but for the example where an entire lifetime of video content is stored on a single disk, it would take at least another lifetime. Instead of attempting to solve the representation mismatch manually, a lot of research is directed at automatically solving the problem.

1.4 Information retrieval

Information Retrieval (IR) is the discipline of finding information in collections. Text retrieval, image retrieval and video retrieval are subfields of IR. Typically research on automatically solving the representation mismatch is done in image and video retrieval. For text retrieval, in general both the query and the collection are text-based so that there is no representation mismatch.

1.4.1 Image retrieval

In Content Based Image Retrieval (CBIR), images are retrieved from a collection of images based on an index that is generated by automatically analyzing the content of the images. Mostly the images are retrieved by keyword/key-phrase queries or by *query by example*. In the query by example task, images are retrieved that contain similar content as an example image that is used as query. Although the query images and the images in the collection are of the same modality, it is not possible to compare them directly. The representation of both query and collection need to be altered. In order to compare the images, for each image a mathematical model, or *signature*, is created. This signature contains low-level information about the picture such as shape, texture or color information.

Directly comparing signatures is possible for the query by example task when the results should be *visually* similar, but unfortunately when the queries are *conceptual* of nature ('Find a picture of a beach', or: 'Find the Tower of Pisa') the signatures do not provide enough information to solve the representation mismatch. This was shown at a recent CBIR benchmark. These benchmarks, where participants all run their system on the same task, have been initiated to compair the performance of CBIR systems [MGMMC04]. Examples of such benchmarks are *the Benchathlon* and *imageCLEF*. ImageCLEF is part of the Cross Language Evaluation Forum (CLEF) benchmark. The benchmark of 2006 contained two retrieval sub-tasks that were both executed on the same archive. This archive contained general, real-life photographs annotated with captions containing high-level conceptual information (such as 'sea' or 'Tower of Pisa') [CGD$^+$07]. In the first task, participants were allowed to incorporate information from both the caption of each image and the images themselves into their systems to perform CBIR. In the second task, which was of the so called type query-by-example, for each query three example images were provided, but the captions could not be used. The systems performed consistently better on the first task than on the second task, illustrating that for CBIR tasks, it is hard to solve the representation mismatch solely on the basis of low-level features [CGD$^+$07].

In an attempt to partially solve the problem, a lot of CBIR systems work semi-automatically. After providing an initial query and reviewing the results, the user can refine the query and in this way, express his interpretation of the meaning of a picture. An overview of recent CBIR research can be found in [DJLW06]. The first serious CBIR applications date from the early 1990s [MGMMC04]. More recent public examples of CBIR technology are the *Riya*[2] search engine and the *Automatic Linguistic*

[2]www.riya.com

Indexing of Pictures - real-time (ALIPR) automatic annotation system [LW06].

1.4.2 Video retrieval

Where image retrieval focuses on stand alone images, in content-based *video retrieval*, the goal is to support in searching video collections. For this purpose, various methods of abstracting information from the video recordings are employed. Because video consists of a sequence of still pictures that are played rapidly after each other, in video retrieval a lot of image retrieval techniques can be re-used, but also other techniques are used such as for example detecting scene changes or recognition of text that is edited in the video (like people's names). Because most videos contain people speaking, it is also possible to use speech as a source of information.

Exploiting speech information can improve video retrieval systems considerably as shown in the TREC[3] Video Retrieval Evaluation (TRECVID) which is a yearly benchmarking event for video retrieval. In 2006, there were 76 submissions from 26 different groups for the fully automatic search task [KOIS06]. The eight best submissions all used information automatically obtained from speech [CHE$^+$06, CHJ$^+$06, HCC$^+$06, CNZ$^+$06].

Comparable to the caption information for content based image retrieval, information from speech helps in solving the representation mismatch considerably. The text that the two sources consist of, is probably not precise enough to contain *all* needed information, but the information that it *does* carry is represented nicely in the same format as the query. Because of the ambiguous nature of language, the fact that sometimes the meaning of sentences can be interpreted in more than one way, it is still possible that a mismatch between the information need of the user and the information in the text sources occurs, but judging from the results of the benchmarks, the gap is smaller than when solely using the other information sources.

1.4.3 Spoken document retrieval

Speech, in most multimedia archives, is a rich source of information for solving the representation mismatch. Sometimes it is even the only reliable source of information. Radio shows or telephone recordings do not contain any video. They might contain some music or sound effects, but generally for those examples most information is in the speech.

Spoken Document Retrieval (SDR) is a subfield of information retrieval that solely focuses on the use of speech for retrieving information from audio or video archives. In the most widely studied form of SDR, in order to solve the representation mismatch the speech is automatically translated into written text by Automatic Speech Recognition (ASR) technology. The output of this process, *speech transcriptions*, can be used in a retrieval system (see figure 1.2). The transcriptions contain the exact time that each word is pronounced so that it is possible to play back all retrieved words. This method is similar to the earlier mentioned example of an index in a book where the page

[3]TRECVID is a video retrieval evaluation event. It is part of the Text REtrieval Conference (TREC) series.

number of each word is stored. Both such an index and speech transcriptions are often referred to as *metadata*. Metadata is data about data. In the speech transcription case, the words and the timing information provide information about the actual data, the audio recordings.

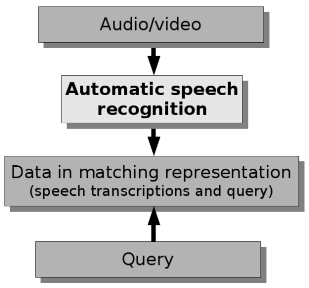

Audio/video
Automatic speech recognition
Data in matching representation (speech transcriptions and query)
Query

Figure 1.2: *Solving the representation mismatch between content and query in an SDR system. The speech from multimedia documents is translated into written speech transcriptions by the ASR component. As the query is already formulated in written text, it does not need to be translated and can be used directly by the retrieval component to find relevant video fragments.*

If the speech transcriptions would always contain exactly what is being said, the performance of the text retrieval system would be equally good as when searching in written text. In general ASR systems are not perfect and any word that is recognized incorrectly, potentially introduces errors in the retrieval component. This was illustrated by the *cross recognizer retrieval* task during the seventh Text REtrieval Conference (TREC-7) in 1998 organized by the National Institute of Standards and Technology (NIST). Participants of the benchmark evaluation used speech transcriptions of varying quality to perform text retrieval. The results showed that although the speech transcriptions didn't have to be perfect in order to obtain good retrieval performance, there was a significant correlation between the quality of the transcriptions and the performance of the retrieval system [GAV00]. This illustrates that the success of an SDR system is highly depending on the performance of the ASR component.

1.5 Automatic speech recognition for SDR

There is a number of methods that can be deployed for recognizing speech for SDR purposes. The most widely used methods are keyword spotting, sub-word recognition and large vocabulary continuous speech recognition.

1.5.1 Keyword spotting and sub-word recognition

One of the earliest speech recognition methods for SDR was *keyword spotting* [RCL91]. In this form of automatic speech recognition, the system does not translate all speech into words, but instead it tries to locate members of a pre-defined list of keywords. The collection index is limited to this list and therefore the retrieval component is only able to find information in terms of these keywords. The main advantage of keyword spotting compared to other approaches is that it is computationally inexpensive.

The drawback of keyword spotting is that only a pre-defined set of keywords can be used for retrieval. The approach in [JY94] solved this problem. First, an automatic phone recognizer processes the audio documents and creates a special phone representation called a lattice. At search time, the query is first translated into a sequence of phones and then all lattices are searched for the phone sequence with a method called *phone lattice scanning*.

Another way to make search for an unlimited set of words feasible is *sub-word recognition* [SMQS98]. For sub-word recognition, a speech recognizer is built that can recognize small speech units or *sub-words* such as syllables or phones. These sub-words are used to create an index. During retrieval, the query words are translated into sequences of sub-words and the index is searched for identical sub-word patterns. Note that this approach is similar to phone lattice scanning. The two approaches differ in that for sub-word recognition an index is created and the documents are not searched directly for phone patterns. Although a bit more complex than keyword spotting, sub-word recognition and phone lattice scanning are still relatively computationally inexpensive. They have the advantage that it is possible to find information not just for a limited set of keywords but for any word or sequence of words.

1.5.2 Large vocabulary continuous speech recognition

The most common form of ASR used for spoken document retrieval nowadays is *Large Vocabulary Continuous Speech Recognition* (LVCSR). Similar to the sub-word recognition approach, LVCSR recognizes small acoustic units using statistical models called *acoustic models*. Typically for LVCSR, these units are phones and not syllables. During the recognition process, the phones are combined into words with the aid of a *pronunciation dictionary* and a *language model*. The dictionary maps sequences of phones to words while the language model can be regarded as a grammar based on statistics. Given a particular context, a language model can determine how likely it is that a certain word is uttered. The language model often helps the system to make correct decisions even if the acoustic models are causing errors. Because of this, in general, LVCSR systems can output higher quality transcriptions than sub-word systems. The downside of LVCSR systems is that in order to map the phones to words and to be able to use a language model, a fixed set of words, a *vocabulary*, needs to be defined. For such LVCSR systems, words that are not in this vocabulary can never be recognized correctly. Each word that is added to the vocabulary will increase the computational efforts during recognition, but fortunately, with todays computer power, the number of words that can be put in the vocabulary is very high. Vocabularies of $50,000$ words to more than $300,000$ words are no exception, reducing the number of out-of-vocabulary words to a minimum.

1.5.3 Sub-word recognition or LVCSR?

Sub-word recognition systems have two advantages over large vocabulary continuous speech recognition. Sub-word recognition systems are computationally inexpensive and they are not restricted by a vocabulary. The advantage of LVCSR systems is that they can create transcriptions with high accuracy thanks to the additional information of their language model. For recognition tasks such as recognizing speech in broadcast news shows, where the number of out-of-vocabulary words can be reduced to a minimum and good use of the language model can be made, in general, SDR systems based on LVCSR will outperform sub-word based systems. For tasks where it is more difficult to minimize the number of out-of-vocabulary words, it is hard to predict which one is the better choice.

In a recent study on speech recognition in the meeting domain, the performance of a sub-word recognition system was compared to an LVCSR system [SSB+05]. In this study, the best retrieval results could be obtained with LVCSR, but the experiments were slightly in favor of the LVCSR system because all query words were present in the vocabulary of the LVCSR system to avoid the out-of-vocabulary problem. Because of the out-of-vocabulary problem in LVCSR systems, some research groups prefer to use sub-word based systems [Li08]. Other groups use hybrid systems that apply a combination of LVCSR and sub-word techniques in order to solve the out-of-vocabulary problem [MMRS08, ESS08, SFB08].

Given the target domains in this research (next to news also meetings and historical data from the cultural heritage domain) it is expected that the benefit of having additional information from language models is significant. As sub-word techniques could always be applied in a hybrid fashion the LVCSR approach is chosen as starting point.

1.5.4 Segmentation and clustering

Segmentation and clustering modules are part of most LVCSR systems. A segmentation module is responsible for segmenting speech input in smaller chunks that can be processed by the recognizer directly. Often the segmenter also filters out non-speech such as silence, lip-smacks, laughter or even tunes or sound effects. The clustering module is used to group together segments with similar characteristics. Obvious characteristics to cluster on are audio channel (broadband/telephony) or gender. Some clustering systems, called *speaker diarization* systems, are able to cluster speech fragments from individual speakers. Using the clustering information the recognizer can process each cluster optimally. For example, special gender dependent models (see section 1.5.6) can be applied when gender information is available or model adaptation techniques can be applied for each separate speaker when a speaker diarization system has been used.

1.5.5 Computational constraints

A very important characteristic of LVCSR systems for spoken document retrieval is that they are not required to produce transcriptions instantly in real-time. Because of this, it is possible to apply high quality algorithms for segmentation, clustering and ASR that process an entire recording before outputting the result. For other types of ASR systems, such as for example dictation systems, this approach is not possible as for these systems results should be available instantly and without intrinsic delay.

Although for SDR, the LVCSR system does not require to generate transcriptions without any delay, it is not the case that they are not bound to any computational constraints at all as such systems may need to process archives of hundreds of hours of material or more in reasonable time.

1.5.6 Statistical models

Whatever types of ASR, segmentation or clustering methods are chosen, most of them are based on statistical methods that require statistical models to take classification or recognition decisions. Segmentation systems often use a speech model and a non-speech model to distinguish between speech and non-speech events, while a lot of speaker diarization systems make use of unified background speaker models, models that represent speech from an 'average' speaker that can be adapted to each individual speaker in the audio. Keyword and sub-word recognizers use acoustic models to determine which sub-word units are most likely pronounced, while LVCSR systems also use language models to determine how likely a word is pronounced given a certain context. In figure 1.3, an example ASR system including all its statistical models is shown.

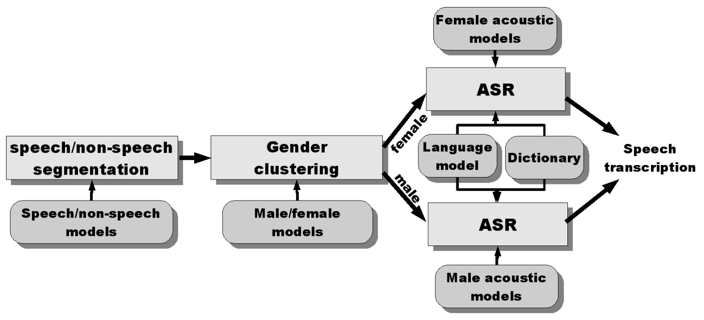


Figure 1.3: *Example of a basic ASR system. The segmentation module filters out all non-speech while the clustering module determines the gender of the speaker in each segment. The recognizer uses either male or female acoustic models to recognize the speech. All components make use of statistical models that are created using example data.*

The acoustic models and language models are created by obtaining statistical information from example data. Machine learning techniques such as the Expectation-Maximization method [DLR77] are used to slowly 'teach' the system how the models should look like using these example or *training* data. The properties of the models and thus the performance of the ASR system depend directly on the nature of the training data. For example, when speech from telephone conversations is used as training data, the system will be good in recognizing speech recorded from telephone lines, but it will probably perform poorly on speech recorded in a television studio because the audio conditions with which the acoustic model was trained do not match the conditions in the studio. Therefore it is important to choose training data wisely when creating an ASR system.

1.5.7 LVCSR for broadcast news recordings

At the department of Human Media Interaction (HMI) at the University of Twente, a spoken document retrieval system for Dutch broadcast news shows was built [Ord03]. This publicly accessible demonstrator[4] processes broadcast news shows on a daily basis. The models for its LVCSR component are trained using a newspaper text corpus of in total some 400M words and an acoustic training set of approximately 20 hours of broadcast news speech. The performance of the LVCSR system is conform the achievements at TREC-7 reported in [GAV00] and the quality of the generated speech transcriptions indeed is high enough to adequately unlock the news archive.

1.5.8 Robustness

In general, a computer application is considered *robust* if it performs well under all conditions, including unusual and unpredictable conditions. It is robust when it is able to deal with unpredicted input with only minimal loss of functionality. This definition of robustness is also valid for LVCSR systems [JH95]. In this thesis, an ASR system is considered robust if it is able to perform well under various audio conditions and for various applications without the need of manually re-tuning the system. A system is robust if it is able to unravel any kind of audio recording that you surprise it with.

In [HOdJ05] it is illustrated that in this sense, the Dutch broadcast news system is not robust. It was used to unlock a multimedia archive of interviews with the famous Dutch novelist Willem Frederik Hermans[5] (WFH). Without any changes to the models, the quality of the generated speech transcriptions was too low for effective SDR. Even after adjusting the models on the limited available acoustical and textual data that could be considered typical for WFH, the ASR performance was not as high as in the broadcast news domain. This illustrates that when speech with new unseen characteristics needs to be recognized, new models trained on data with these same characteristics are needed. New (large) training collections need to be bought or created which is a time consuming and therefore expensive task.

1.5.9 Towards LVCSR for surprise data

As said, the system for Dutch broadcast news is able to adequately unlock a Dutch broadcast news archive. A one time effort was needed to create the statistical models, but once available, they can be deployed for the fully automatic transcription of

[4]`http://hmi.ewi.utwente.nl/showcase/broadcast-news-demo`

[5]The SDR system is part of an internet portal: `http://www.willemfrederikhermans.nl`

broadcast news[6]. As the WFH example illustrates however, re-tuning the models is required as soon as the characteristics of the audio changes. It shows that with a broadcast news system, it is not possible to generate high quality speech transcriptions for any arbitrary set of *surprise data*, a data set for which the audio conditions and the topics of conversation are a surprise to the system. The problem of manually creating speech transcriptions for each multimedia document is shifted to developing relevant models for each multimedia archive.

Sometimes it is worth the effort to collect new training data and create new models. Governments are willing to spend money on technology for the monitoring of telephone lines and companies might be willing to invest in automatically creating minutes or monitoring commercial campaigns of competitors, but in a lot of other cases training data are not available and creating new data is simply too expensive. This problem brings up the question: 'Is it possible to develop a system that can handle any kind of surprise data?'. In other words: 'Is it possible to automatically or semi-automatically adapt existing ASR systems to ASR systems for new application domains?'. In the case of LVCSR systems that use two kinds of statistics (acoustic models and language models), this question can be split up into two questions:

- How can a LVCSR system be made robust against new unseen audio conditions?
- How can a LVCSR system be made robust against new unknown language structures with potentially new words?

The audio conditions are determined by numerous factors during the creation and storage of the audio. Not only background noise influences the audio quality, but also other factors such as the microphones used for recording, the medium used for storage and the location of the recording. Audible non-speech events also need to be considered. Television shows may contain audible non-speech such as laughter and applause or even (background) music and sound effects. Even recordings of meetings that can normally be considered as 'clean', data might contain audible non-speech such as papers shuffling, doors slamming or even colleagues playing table football in another room.

When changing the application domain, it is likely that the language used in the audio recordings will change as well. Broadcast news lingo is different from the lingo in corporate meetings. The sentence structure will differ and also the kind of words that are being used. These changes require a new vocabulary and language model, but finding enough text data for training these models is often a problem.

The goal of the research described in this thesis is to answer the first question and to demonstrate the proposed solutions using the Dutch broadcast news LVCSR system developed at HMI as starting point. Although the second question is equally important, changes in language affect the system only in one place: where the language model is needed. It is expected that adapting a system to be robust against changing audio conditions will require changes in all system components of which most

[6]Note that if the system is deployed for a longer period, some continuous effort is needed to keep the statistical models up-to-date.

are needed before the language model is used (for example removing sound effects). Therefore it is justifiable to address the first problem before the second one is tackled.

1.6 About this thesis

In this chapter an introduction has been given into one of the problems that needs attention when applying automatic speech recognition to spoken document retrieval of collections with unknown characteristics. This problem will be turned into a research agenda which distinguishes the various research goals and development requirements underlying the PhD work presented here.

1.6.1 Research goals

The goal of the research described in this thesis is to address the fundamentals of an ASR system that is able to process audio with new unseen characteristics. As described in the previous section, the problem of processing audio with unseen audio conditions is that these conditions are likely not to match the conditions in the training data. LVCSR systems using statistical models trained on the training set and parameters tuned on this set will perform suboptimal because of this mismatch. This problem is observed in each of the three subsystems that can be distinguished in most LVCSR systems: *segmentation*, *clustering* and *ASR*. Therefore, for each of these subsystems, research will focus on answering the main research question:

- 'How can a LVCSR system be designed for which all statistical models and system parameters are insensitive to potential mismatches between training data and target audio?'

There are two approaches in solving the mismatch problem: the models, parameters or the data can be normalized so that the mismatch is reduced, or a system is developed that does not need models or parameters created using training data at all. Under the first approach, in general the mismatch can be reduced but not completely prevented. Therefore, the second method of overcoming the need of training data and therefore removing the mismatch altogether, is appealing. In many cases though, statistical methods that require training data simply outperform other methods, even when there is a data mismatch. Therefore, it is not enough to ask how a system can be created that reduces or removes the data mismatch, but the following question needs answering as well:

- 'What is the performance of the proposed system compared to the state-of-the-art?'

In order to answer this question, the proposed system and isolated parts of the system are evaluated on international benchmarks. This way the system performance can be compared to the performance of other state-of-the-art systems that processed the same task. The results are not only used to determine the relative performance, but also to identify the weak steps and to find out which steps can be improved most.

The procedure is the same for each of the three subsystems. In the sequel of this section, first a method will be proposed that reduces or removes the mismatch between training data and the data that is being processed. Next, the method will be evaluated on a well known benchmark so that its performance can be compared to that of others. Third, an analysis is performed that identifies the weaknesses of the method. These steps can then be repeated and a new method can be proposed for which the known flaws are addressed.

Segmentation

In general audio may not only contain speech but also various kinds of other sounds. These non-speech fragments such as background noise, music or sound effects need to be separated from the speech segments. A common method is to model speech and each of the audible non-speech events so that they can be identified during segmentation. This approach requires that the kind of non-speech encountered during segmentation is known, which is often not the case. Also this method requires that training data is representative for the data that needs to be segmented. In chapter 4 it will be shown that even when no audible non-speech is present in a recording, the system performance will drop significantly if the statistical models are trained on mismatching data. Therefore, the two following questions need answering:

- 'How can all audible non-speech be filtered out of a recording without having any prior information about the type of non-speech that will be encountered?'
- 'How can the system perform speech/non-speech segmentation without the use of statistical models based on training data?'

Speaker clustering

Speaker clustering systems suffer from the same problems as segmentation systems. Because speakers are not known beforehand, it is impossible to train perfect statistical models for them. Per definition the data to train models on will not be a perfect match to the data that contain speech of the actual speakers. Therefore the following question will be addressed:

- 'How can a speaker clustering system be designed that does not require any statistical models built on training data?'

In chapter 5, a system design will be proposed that can do this. A disadvantage of this system is that it is computationally expensive for long recordings because it requires pair-wise comparison of all fragments of the recording. The longer the recording, the more of these comparisons are needed. Therefore in a second iteration, a new system will be proposed that addresses the question:

- 'How can a speaker clustering system be designed that is able to process long recordings with reasonably computational effort?'

Automatic speech recognition

From a software engineering point of view, the ASR subsystem is the most complex of all three subsystems. This is the reason why in the chapter about ASR, chapter 6, a number of *development* issues will be addressed. One of these development problems is how to implement a decoder that can easily be modified for research purposes, but that nevertheless can operate with reasonable computational requirements. One technique that is very helpful in managing the computer resources during decoding, is Language Model Look-Ahead (LMLA). Unfortunately, it is not straightforward how to use this technique with the system architecture that was chosen in order to fulfill the development requirements and therefore the following research question will be addressed:

- 'How can full language model look-ahead be applied for decoders with static pronunciation prefix trees?'

For decades, research has been performed on making automatic speech recognition more robust in various kinds of ways. For example, numerous methods have been developed for noise reduction, for robust feature extraction in noisy environments or for creating more uniform acoustic models. These methods all aim at the creation of systems that are insensitive to the potential mismatch between training data and target audio and address the question:

- 'Which methods can be applied to make the decoder insensitive for a potential mismatch between training data and target audio?'

Unfortunately it was not feasible to implement and experiment with all known methods. Instead, a selection of methods was picked that proved itself in various international benchmarks. In chapter 2, an overview of these methods will be given.

1.6.2 Development requirements

Development and implementation of the ASR system are important parts of the work described in this thesis. It must be easy to implement new ideas quickly and transparently into every part of the software, so that they can be validated with experiments. Such an environment makes it easy to replace parts of the system with alternative implementations. Although the goal of this research is to create a system that is as robust as possible for new audio conditions, no concessions shall be made to the readability and transparency of the resulting software.

The flexibility of the framework is obtained by developing a modular software architecture. Each module performs a separate task and interacts with other parts of the software using well defined and transparent interfaces. The modules are built so that it is possible to re-use them in another setting or replace them by alternative implementations. Especially for the following three aspects of the design, the modularity of the design is very important.

A strict distinction needs to be made between the algorithms that are independent of what kind of data will be used and the part of the system that changes whenever the precise task changes. For example, a system may be created for both the Dutch language and for English. All system parts that are different for the two languages need to be strictly divided from the language independent parts. The language dependent parts will be stored in binary statistical model files and defined in configuration files, but not in source code. This distinction makes it possible to apply the overall system to various languages without having to adjust any source code.

Second, the software in the framework must be modular with respect to functionality. Algorithms for handling language models may interact with other system parts, but this must happen through well defined interfaces. It must be possible to replace a component such as the software that handles the language model by an alternative implementation without having to adjust any of the other components. This type of modular design will make it possible to perform research on one particular topic (for example on acoustic modeling) and create and test various methods without the need of touching other parts of the source code.

Third, the framework needs to be set up so that it is possible, and easy, to reuse general purpose components. For example, it must be easy for all algorithms implemented in the framework to make use of a single component for Gaussian mixture PDFs. As not all algorithms will input feature vectors of the same dimensionality, the GMM component needs a flexible interface. This type of modular design will make it possible to quickly implement various applications that are based on similar building blocks.

1.7 Thesis outline

In the next chapter, an overview of the current state of the art in large vocabulary continuous speech recognition will be given. A number of existing systems that scored above average on recent benchmark evaluations will be described and commonly used techniques are discussed. In chapter three, the approach taken to create a robust system for unknown audio conditions is discussed and an overview of the proposed system is provided. This system consists of three subsystems: the speech activity detection subsystem, the speaker diarization subsystem and the automatic speech recognition subsystems. These three subsystems will be discussed in-depth in chapters four, five and six. The evaluation of each individual subsystem is presented in these three chapters. In chapter seven, the overall system evaluation is described. In chapter eight, the conclusions are summarized and directions for future research are suggested.

CHAPTER 2

STATE-OF-THE-ART IN ASR

In this chapter an overview is given of today's state-of-the-art in large vocabulary continuous speech recognition. The overview is given to provide a foundation for the following chapters and to formulate definitions from existing work. It is not intended to provide the ASR history in full, but instead only the techniques needed in the following sections are described and pointers to more information on the various topics are given.

In this chapter, first the statistical methods that almost all state-of-the-art ASR systems are based on will be discussed. Next, an overview is given of key concepts underlying the systems that participated in the major LVCSR benchmark events. The final three sections of this chapter discuss segmentation, speaker diarization and automatic speech recognition.

2.1 Fundamentals

The task of a speech recognizer is to find the most probable sequence of words given some audio represented as a sequence of acoustic observations O. This statistical classification task can be formulated as a search for the maximum a-posteriori probability $\hat{W}$ over all possible sequences of words W. Using Bayes' theorem this search can be expressed as:

$$\begin{aligned}\hat{W} &= \operatorname*{argmax}_W P(W \mid O) \\ &= \operatorname*{argmax}_W \frac{P(O \mid W) \cdot P(W)}{P(O)} \\ &= \operatorname*{argmax}_W P(O \mid W) \cdot P(W)\end{aligned} \tag{2.1}$$

Because $P(O)$ is the same for each sequence of words and will not influence the search for $\hat{W}$, it can be ignored. The remaining likelihood $P(O \mid W)$ is calculated with the use of acoustic models and the prior $P(W)$ with a language model. As mentioned in section 1.5.2, most recognizers use phones as the basis for the acoustic models and a dictionary is used for mapping words to sequences of phones. Often Hidden Markov Models (HMM) are applied to model the phones. In turn these HMMs make use of Gaussian Mixture Models (GMM).

2.1.1 Feature extraction

The first step in obtaining the sequence of acoustic observations O is to convert an analog audio signal into a digital representation. During this analog-to-digital conversion, the amplitude of the signal is measured at fixed time intervals and translated to a floating point number. Because the information in this sequence of numbers is highly redundant, it is transformed into a reduced representation so that the relevant information is maintained but the data is less redundant. This step is called *feature extraction.*

First, a short-time spectral analysis is performed on the amplitude sequence. This spectral information is then used as input for a filter that modifies the information according to a model for human hearing. Two commonly used methods for this are Mel Filtered Cepstral Coefficient (MFCC) [DM80] analysis and Perceptual Linear Predictive (PLP) analysis [Her90]. Both methods output a series of vectors. In a final step, the first and second order derivatives are often concatenated to these *feature vectors.*

A more extensive discussion on feature extraction can be found in [JM00]. The MFCC and PLP algorithms are described in depth in [YEH$^+$95].

2.1.2 Hidden Markov Models

The statistical model most often used to calculate the likelihood $P(O \mid W)$, is the Hidden Markov Model (HMM). An HMM consists of a finite number of *states* that are connected in a fixed topology. The input of the HMM, the feature vectors, are called *observations.* Each HMM state can 'emit' an observation o_i from the observation sequence $O = (o_1, o_2, ..., o_T)$ with a certain probability defined by its *Probability Distribution Function* (PDF). The first observation must be emitted by a state that is defined to be one of the *initial states.* After this observation has been processed, one of the states that is connected to the initial state is chosen to emit the next observation. The probability that a particular transition from one state to another is picked, is modeled with the *transition probability.* The sum of all outgoing transition probabilities of each state should be one, so that the overall transition probability is also one. Eventually all observations are emitted by a state that is connected to the state that emitted the previous observation (if the HMM contains self-loops this can actually be the same state) and finally, observation o_T should be emitted by one of the *final states.* By taking multiple paths through the model, identical sequences of observations can be generated. The actual path taken to create a specific state sequence is unknown to a theoratical observer and therefore this type of Markov Model is called a 'Hidden' Markov Model.

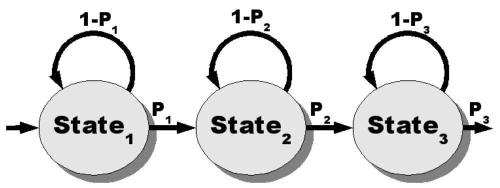


Figure 2.1: *A typical Hidden Markov Model topology for modeling phones in ASR. This left-to-right topology contains three states that are each connected to their neighboring state.*

Figure 2.1 is a graphical representation of a typical HMM topology used to model phones. It consists of three states $State_1$, $State_2$ and $State_3$, and each state is connected to itself and to the following state. $State_1$ is the only initial state and $State_3$ is the final state.

Three problems arise when using HMMs: the evaluation problem, the decoding problem and the optimization problem. The evaluation problem is the problem of finding the probability that an observation sequence was produced by a certain HMM. The decoding problem is the problem of finding the *most likely* path of state transitions given that an observation sequence was produced by a certain HMM. The optimization problem is the problem of optimizing the parameters (the transition probabilities and the PDFs) of a certain HMM given a set of observation sequences. The decoding problem can be solved using the Viterbi algorithm and the optimization problem with the Expectation Maximization (EM) theorem. Both algorithms are used extensively in automatic speech recognition. EM is used to train the HMM parameters and Viterbi is used for recognition of the audio.

An in depth description of Hidden Markov Models and the use of Hidden Markov Models in ASR can be found in [Jel97] and [JM00].

2.1.3 Gaussian Mixture Models

In ASR, the probability distribution functions of the HMMs are often Gaussian Mixture Models (GMM). A GMM is a continuous function modeled out of a mixture of Gaussian functions where the output of each Gaussian is multiplied by a certain weight w. The Gaussian weights sum up to 1 and the Gaussian functions themselves are defined by their mean vector μ and covariance matrix Σ. The covariance matrix Σ is mostly restricted to diagonal form because this simplifies the decoding and training process considerably. The following formula defines a GMM with i Gaussians for input vectors of n dimensions where $(o-\mu_i)^T$ is the transpose of $(o-\mu_i)$:[1]

$$f(o) = \sum_i w_i \frac{1}{\sqrt{(2\pi)^n\,|\Sigma_i|}} e^{-\frac{1}{2}(o-\mu_i)\Sigma_i^{-1}(o-\mu_i)^T} \tag{2.2}$$

[1]Note that in order to obtain the true probability of a certain observation using a GMM, the integral of $f(o)$ by the integration interval of the input variable o should be taken.

The mean vector, covariance matrix and weight of each Gaussian in the GMM need to be set so that $f(o)$ is maximal for the class of observations that the GMM represents (a certain phone). This optimization is done simultaneously with the optimization of the HMM transition probabilities using the EM theorem.

Instead of using one GMM as PDF, some systems use multiple GMMs to calculate the observation emission probability. In such a *multi-stream* setup, the PDF output is the weighted sum of these GMMs. Each stream in a multi-stream GMM can be trained with features of its own type. For example, one stream can be using MFCC features while a second stream uses PLP-based features.

2.1.4 Viterbi

The Viterbi algorithm is used to solve the HMM decoding problem of finding the most likely path through an HMM given a sequence of observations. The Viterbi algorithm also provides the actual probability given this most likely path. This algorithm is nicely described in [JM00]. A very short explanation will be given here, followed by a convenient implementation method of Viterbi, the token passing algorithm.

Viterbi using a matrix

The optimal path through an HMM and its corresponding score can be calculated using a matrix with one row for each state s in the HMM and one column for each time frame t. Each cell in the matrix will be filled with two types of information: the maximum probability of being in state s at time t and the state at time $t-1$ from which the transition was taken to obtain this maximum probability.

The matrix is filled as follows. First, all cells in column t_0 that correspond to one of the starting states will be set to one and the remaining cells to zero. Then, each cell in the next column will be filled with the new score of value v:

$$v(s_x, t_i) = \underset{S}{\operatorname{argmax}}\, v(S, t_{i-1}) \cdot P_S(o_t) \cdot P_t(S, s_x)$$

where P_S is the PDF of state S and $P_t(S, s_x)$ is the transition probability from state S to state s_x. The number of the state that resulted in the maximum score is stored as well.

The last column that represents the final time frame T, contains all probabilities after emitting all observations. The maximum value of all cells corresponding to one of the final states is the probability of the most likely path. The path itself is obtained by backtracking all state numbers back to the first column.

Viterbi using the token passing algorithm

A convenient method of calculating the Viterbi score is using the token passing algorithm [YRT89]. In this algorithm, each state in the HMM is able to hold a so called token, containing the optimum probability of being in that particular state at a certain time t_i. At t_0, only the initial states will be provided a token. At each time frame, these tokens, with initial value one, will be passed to all connected states. Before doing this, the value of the token is updated and also the state it came from is marked on the token. If a state has more than one outgoing transitions, the token will be split and passed to all connected states. The new value of the token v in state s_y coming from state s_x will be:

$$v(s_y) = v(s_x) \cdot P_{s_x}(o_t) \cdot P_t(s_x, s_y)$$

where P_{s_x} is the PDF of state s_x and $P_t(s_x, s_y)$ is the transition probability from state s_x to state s_y. It is possible that two tokens arrive at the same state at the same time. In this case, the token with the highest value will be obtained and the other token will be discarded. At time T, from all tokens that are in one of the final states, the token with the highest value is chosen. The value of this token is the probability of the most likely path through the HMM. The actual path is marked down on the token itself[2].

2.1.5 Acoustic models

As mentioned earlier, most LVCSR systems use phones as the basis for acoustic modeling. A fixed set of phones is defined and an HMM with the topology of figure 2.1 is used to model each phone. In [SCK$^+$85] it was shown that because the pronunciation of phones is affected significantly by their neighboring phones, recognition performance increases when the pronunciation context of each phone is taken into account. Instead of training one model for each phone, a model for each phone with unique left and right neighboring phones, called a *triphone* can be created. For a phone set of N phones, this means that a set of N^3 *context-dependent* phones need to be trained. Unfortunately, this causes a data scarcity problem as not enough training data will be available for most of these contexts. In [SCK$^+$85] this problem was solved by using both context-independent phone models and context-dependent phone models and weight them by a factor depending on the amount of available data.

In [YOW94], a tree-based method was used to solve the data insufficiency problem. This effective clustering method uses a binary phonetic *decision tree* to cluster phone contexts, so that each cluster contains sufficient training data for the HMM. A list of questions about the type of phone to the left or to the right of the phone that needs to be trained is used for this clustering. Typical questions are: ‘is the phone to the left a vowel?’, ‘is the phone to right a fricative?’ or: ‘is the phone to the left the m-phone?’. The following algorithm is used to create a decision tree using this list of questions:

- First, an initial data alignment is created so that all observations are assigned to one of the three HMM states.
- For each state, the data is placed in the root of the decision tree and the likelihood of the entire data set using a single Gaussian function is calculated.

[2]typically as a linked list of state numbers

- The data is split into two according to the question that increases the total likelihood the most after a single Gaussian function is trained for each of the two new clusters.
- Splitting the clusters is repeated as long as each cluster contains a minimum amount of data and the likelihood increases more than a certain threshold.

Using this algorithm, a tree such as shown in figure 2.2 can be created. For each of the leaf clusters, an HMM state is trained (the GMMs and transition probabilities). During recognition, for each triphone, the state trained on the corresponding cluster (the node reached by answering all the questions in the decision tree for the particular triphone) is used. This means that for all triphones, also the ones that did not occur in the training set, a model is available.

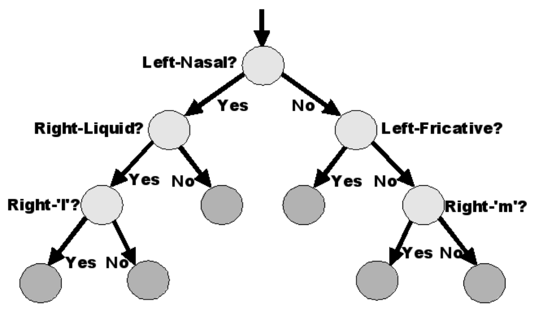


Figure 2.2: *An example of a phonetic decision tree (from [YOW94]). By answering the questions in the tree, for each triphone a leaf node will be found that contains a corresponding model.*

2.1.6 Language model and vocabulary

The a-priori probability $P(W)$ in formula 2.1, where W is a sequence of words $w_1, w_2, ..., w_n$, is calculated using a language model. LVCSR systems normally use a statistical n-gram model as language model. In n-gram models, for each possible sequence of $n-1$ words, the probability of the next word w_i is stored. The a-priori probability calculated using a trigram (3-gram) language model is as follows:

$$P(W) = \prod_i P(w_i \mid w_{i-1}, w_{i-2}) \tag{2.3}$$

The probability $P(w_i \mid w_{i-1}, w_{i-2})$ of all possible word combinations are obtained from statistics of large text corpora. For word combinations of which not enough statistical evidence is found, lower order n-grams are calculated. Multiplied with a 'back-off' penalty, these probabilities can be used instead of the higher order n-grams. Because obtaining these statistics is only possible when the set of possible words is limited and fixed, a *vocabulary* needs to be defined before the n-gram model can be created. In LVCSR systems, vocabularies are defined that consist of more than 50K words. Some systems even use vocabularies with more than 300K words. With these large vocabularies the risk is minimized that words are encountered in audio recordings that do not occur in the vocabulary. Words that are missing in the vocabulary are called *out-of-vocabulary* (OOV) words and in [Ord03] it is shown that for a Dutch broadcast news system, the percentage of OOV words, the OOV rate, for a vocabulary of 65K words is only 2.01%.

2.1.7 Dictionary and pronunciation prefix tree

The pronunciation, in terms of sequences of phones, of each word in the vocabulary is stored in a *dictionary*. This dictionary is needed so that the HMM phone models can be concatenated into word models. Two HMMs with a topology as in figure 2.1 are concatenated by connecting the outgoing transition of $state_3$ of the first HMM to the incoming transition of $state_1$ of the second HMM. The outgoing transition probability of $state_3$ is used as new transition probability (so that the sum of all transition probabilities of $state_3$ remains 1). The word models that are created by stringing phone HMMs together like this are HMMs as well. In fact, it is even possible to create one big HMM out of all the word models by placing the models in parallel and connecting all incoming transitions of $state_1$ of the first phone of each model and all outgoing transitions of $state_3$ of each final phone with a so called non-emitting state. These kind of states do not emit observations, but only contain state transition probabilities [JM00].

Because this single model, containing all words from the vocabulary, is one big Hidden Markov Model, it is possible to use the Viterbi algorithm to solve the decoding problem. Solving the decoding problem, the problem of finding the optimum path through an HMM, will result in finding the most probable pronounced word given the sequence of observed feature vectors. Unfortunately, the number of states in a big model such as this is very high making it computationally expensive to perform Viterbi. Therefore, often a *Pronunciation Prefix Tree* (PPT) is used instead. In a PPT, the word models are not connected in parallel, but instead words with the same initial phones share those phone models as shown in figure 2.3. A Viterbi search through this HMM topology will have the exact same result as when parallel word models are used but because the PPT consists of less states, the search will be computationally less expensive.

2.1.8 The decoder

The application that is responsible for finding $\hat{W}$ in formula 2.1, is often called a *decoder* because it decodes a noisy signal ('noise' from the articulatory system and transmission through the air are added to the word sequence) back into words. It is

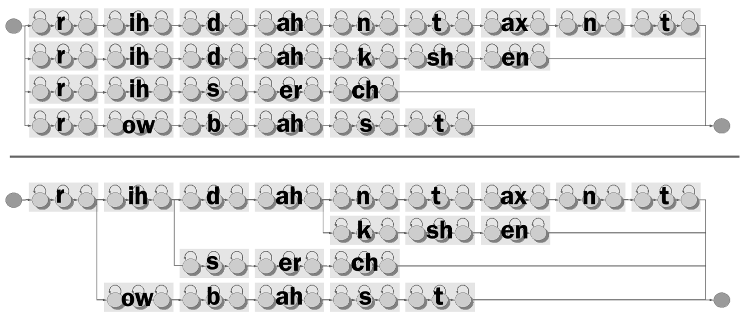


Figure 2.3: *Two HMM topologies representing the four words: 'redundant', 'reduction', 'research' and 'robust'. In the topology at the top, all word models are placed in parallel. In the topology at the bottom, the pronunciation prefix tree (PPT), all words with the same initial phones share the phone models. Although using the Viterbi algorithm, the most probable pronounced word will be the same for both topologies, the PPT requires a lot less states.*

also the application that needs to solve the HMM decoding problem on the pronunciation prefix tree.

The decoder uses a dictionary and HMM phone models to create a pronunciation prefix tree. After feature extraction is performed and the sequence of feature vectors (O) is available, the Viterbi algorithm can be used on this tree to find the first most probable word and its posterior probability $P(O \mid w_1)$. The language model is then used to find the a-priori probability $P(w_1)$.

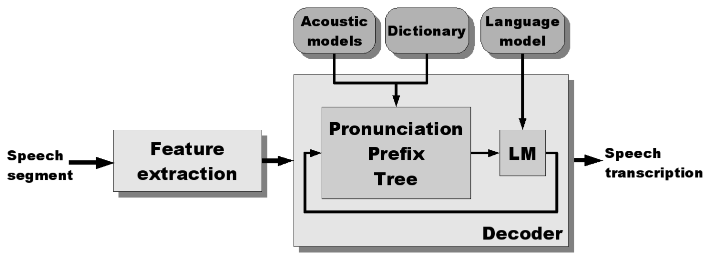


Figure 2.4: *Top-level overview of a decoder. The AM and dictionary are used to create a PPT. In this example, after the LM probability is incorporated to the acoustic probabilities, the word history is stored and the PPT is re-entered at the initial states.*

In order to decode a sequence of words instead of just one single word, the PPT needs to be extended so that it is possible to form unique paths through the HMM of n words when an n-gram LM is used. Without special measures, just connecting the final states of the PPT with the initial states would make it impossible to distinguish between word histories and calculate $P(w_i \mid w_{i-1}, ..., w_{i-n})$. One solution to this problem is to place copies of the PPT on the outgoing state transition of each word and repeat this $n-1$ times. The transitions of the final PPTs can then be connected to the initial states of the corresponding PPT at the final level as is shown in figure 2.5. Another solution to the problem is to simply connect the final states of the PPT with the initial states, but to add the word path history information to each token during Viterbi token-passing. Tokens that do not share the same history will not compete for a place in a state, but can occupy a state simultaneously. This means that instead of a token administration, a token-list administration is needed in each state.

Whatever method is chosen, either tree copying or extending the token administration, the number of possible paths in the resulting HMM will be huge. Therefore, decoders often use the *beam search* method in which the least promising paths are pruned out of the search. This method makes it possible to decode faster, but when the search beam is too narrow and too many paths are pruned away, it is possible that also the best solution is disregarded.

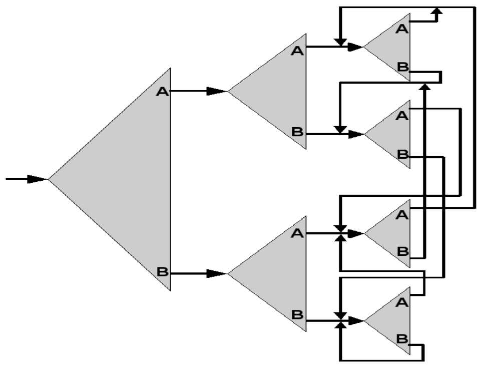


Figure 2.5: *One way of keeping track of word history is to create PPT copies. This is an example of how a trigram topology would look like for a PPT containing the two words A and B. At each arrow in this diagram it is possible to calculate the corresponding LM score.*

Note that connecting PPTs becomes more complex when context dependent phones (see section 2.1.5) are used. For *within-word* dependency, where only the context of phones within a word are used, connecting the PPTs is not affected. But when *cross-word* dependency is applied, where also the context of the first phone of the next word is used, special measures need to be taken. A straightforward method is to replace the initial and final phones of the PPT (of each word) with a set of phones representing all contexts and connect the appropriate phones [FFKW99, DDCW00].

2.1.9 Decoder assessment

The performance of a decoder can be measured in two ways: in degree of speed or degree of accuracy. The standard evaluation metric used to measure the accuracy of a decoder is the *Word Error Rate* (WER). WER is defined as the minimum edit distance between the *reference transcription* of the test material and the output of the decoder, the *hypothesis transcription*, as a percentage of the length of the reference transcription. The minimum edit distance between the reference transcription and the hypothesis transcription is the sum of the number of deletions D, insertions I and substitutions S that are minimally needed to align the words of the reference transcription to the words of the hypothesis. The word error rate is defined as:

$$\text{WER} = \frac{D + I + S}{N_{\text{ref}}} \cdot 100\% \tag{2.4}$$

where N_{ref} is the number of words in the reference transcription. Note that the word error rate can be higher than 100%. For example when the hypothesis contains more words then the reference and all of these words are incorrect. In this case the number of substitutions would be equal to the number of words in the reference and on top of that there would be insertion errors.

The word error rate is often used to compare systems, part of systems or new algorithms. For the development of algorithms, it is common practice to evaluate a system on a test set with and without the proposed algorithm. A *significant* improvement in WER proves that the algorithm functions well. It is very important to determine that two hypothesis are actually significantly different because, especially when the test set is relatively small, it is possible that a decrease in WER is simply due to chance. In [GC89], two significance tests are proposed for ASR: the McNemar's test and the matched-pairs test. It is stated that especially the matched-pairs test is suitable for significance testing of connected speech. With this test, it can be calculated what the probability p is that two hypothesis are the same. If p is very small (typically 0.05, 0.01 or even 0.001), the two hypothesis are considered significantly different. An application that calculates p for two hypothesis is described in [PFF90]. This application will be used in the remainder of this work for performing significance tests.

The speed performance of a decoder is often measured with the real-time (RT) factor. The real-time factor is the time that it took to decode the test material (D) divided by the length of this material (L):

$$\text{RT} = \frac{D}{L} \cdot 100\% \tag{2.5}$$

Note that the real-time factor on its own does not provide sufficient information. The computer system used to run the decoder should be specified as well (speed and type of the processor).

2.2 NIST benchmarks for ASR

The US National Institute of Standards and Technology (NIST) has organized speech recognition evaluation benchmarks since 1987 [Pal03]. In these benchmarks, participants perform ASR on an evaluation set provided by NIST. The results of these benchmarks provide an overview of the state-of-the-art in speech recognition. In this section some of these benchmark events will be discussed.

2.2.1 NIST Benchmark procedures

The procedures for the ASR benchmarks organized by NIST are similar for all events. First, participants receive training data that match the evaluation audio as good as possible. The training audio is manually transcribed for the task under evaluation. For ASR this is a time aligned transcription on word basis. For speaker diarization (see section 2.4) this is a time aligned transcript of which speaker is talking when. The training data can be used to develop a system although it is allowed to use other data as well.

Well before the evaluation dead-line (approximately a month), participants receive the evaluation audio. Participants may not manually manipulate this data. It is expected that the system processes the data fully automatically. The hypothesis files are sent back before the evaluation dead-line and NIST will score the submissions using a manually created reference transcript (see section 2.1.9). The scores are ranked and the result is sent back to the participants together with the reference transcriptions so that they can perform post-evaluation analysis. After the evaluation, NIST organizes a workshop where each participating team can present its results.

Often, a benchmark consists of multiple tasks. There is always at least one main task that each team is required to perform. Sometimes, contrastive tasks are also formulated. In the rich transcription benchmark series of meetings for example, various types of microphones where used for recording. For this benchmark, one contrastive task is to perform ASR using the head-set microphones while the main task is to use the microphones placed on the tables.

2.2.2 Benchmark results

Over the years various tasks have been evaluated by NIST. Figure 2.6 shows the best results for these tasks per year. With each benchmark a trend towards lower word error rates can be seen. This is of course because systems improved, but in general also because over the years more training data have become available for the specific tasks.

From 1996 to 1999 the recognition of Broadcast News (BN) recordings was the main task. In 2003 and 2004 the BN task was repeated, but this time the competing systems needed to perform in less than ten times real time (10xRT) (one hour of audio had to be processed with the use of less than ten hours of CPU power). In 2004 NIST started a benchmark series on tasks in the meeting domain. For this task, that is still running, recorded speech from real meeting room discussions needs to be recognized. This task proves to be difficult because of the unconstrained vocabularies, spontaneous and overlapping speech and the variation in recording conditions.

The systems that participated in these three benchmark series, BN with unlimited resources, 10xRT BN and the meeting benchmarks, are especially interesting to be investigated for this research. The first BN systems because the available training data until 1998 is comparable to the amount of data that is currently available for Dutch. The systems that participated in 2004 for the 10xRT BN benchmark are interesting for this research because they are more state-of-the-art than the 1998 systems and

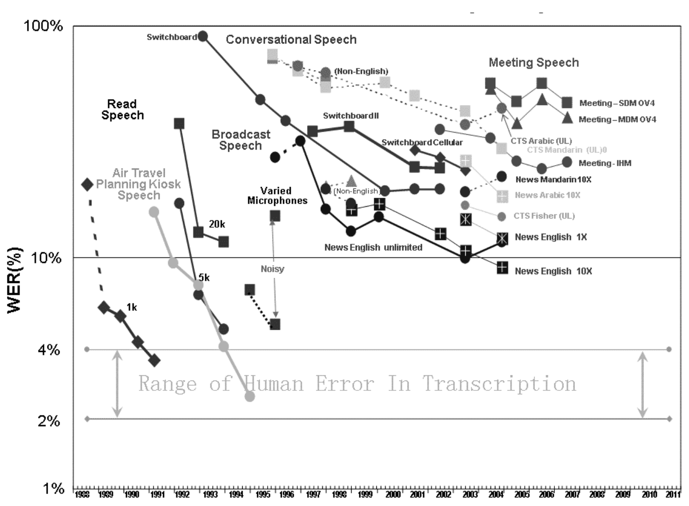


Figure 2.6: *The NIST benchmark test history until May 2007 (from [FA07]). The blue lines represent the lowest obtained WER at each BN benchmark (10xRT,1xRT and unlimited time) and the purple lines represent the best results for the meeting benchmarks.*

therefore provide more information on techniques already applied to make ASR more robust against varying audio conditions. The meeting benchmark is interesting first of all because the most recent systems participated in that benchmark and also because for this task limited amounts of training data are available.

In the remainder of this section a top level overview of the most commonly used system components in the three benchmark series are discussed. This discussion will mainly focus on techniques for acoustic modeling because the goal of this thesis is to prepare a system for unseen audio conditions. In the following sections the discussed components will be investigated further.

2.2.3 The broadcast news benchmark series

Of course none of the participating systems for the BN benchmarks from '95 until '99 were exactly the same. In fact, most systems had a number of distinct features and very interesting experiments were conducted on these features during this benchmark series. However, a number of system characteristics were common between participants. The most common characteristics are shown in figure 2.7. They will be discussed here. In depth information about specific systems can be found in the proceedings of the 1999 DARPA Broadcast News Workshop ([CEG+99, GLAJ99, ZWG99, HCS99, CCE+99, BAHU+99, WLY+99]).

For all participating teams of the 1998 benchmark, the first step of the system was *segmentation*. Audio fragments not containing any speech were removed by the segmentation component. Mostly two categories were distinguished: speech and non-speech such as silence or background noise. Some participants also used a music category to filter out the BN jingles [CEG+99, ZWG99].

After segmentation, most systems employed a *clustering* component. This component clustered segments that showed similar acoustic characteristics. Clustering was done mainly on basis of speakers or groups of speakers [CEG+99, GLAJ99, BAHU+99], and/or gender [GLAJ99, ZWG99] or bandwidth [GLAJ99]. Being able to cluster automatically, enables the possibility of using specific acoustic models such as telephone and broadband models or gender specific models. Automatic clustering also makes it possible to adapt models or normalize feature vectors on meaningful clusters of speech, such as speech from one single speaker.

Although not all systems used the same feature extraction procedures (for example the modulation-filtered spectrogram features in [CCE+99]), most commonly used feature extraction methods were MFCC and PLP. The first *decoding* step was often a rough and fast ASR run. The results of this run were used to adapt the acoustic models (using MLLR, see section 2.5.2) or to normalize the feature vectors (VTLN, see section 2.5.2).

For modeling acoustics, with the exception of one system that used neural network technology [CCE+99], in the benchmark of 1998 all systems used HMMs with GMMs to model the probability distribution functions. Without exception, recognition was performed on phone basis.

Some systems performed an ASR run in order to generate lattices [HCS99, CCE+99] or word graphs [GLAJ99, WLY+99]. Lattices and word graphs are mathematical structures containing multiple possible recognitions. Compared to the initial possible number of word hypotheses, the number of possible recognitions is reduced dramatically. This reduced search space makes it possible for a following decoding run to use higher order language models without running out of computer memory.

Another interesting technique aplied at the broadcast news benchmark series is Recognizer Output Voting Error Reduction (ROVER) [Fis97]. This method that was used by a number of systems [CEG+99, ZWG99, CCE+99], is able to combine the output of multiple recognizers into one single recognition using a voting scheme.

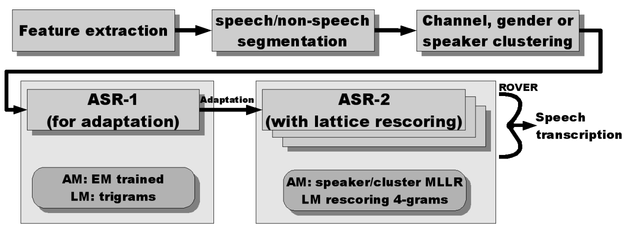


Figure 2.7: *Top level system overview of most systems that participated the BN benchmarks between 1995 and 1999. After segmentation and clustering, two ASR passes are performed. The first pass is used for speaker or cluster adaptation. A third lattice re-scoring pass is sometimes performed with a higher order LM and when multiple ASR outputs are available, ROVER is applied.*

2.2.4 The 10xRT broadcast news benchmark series

The systems used in the 10xRT BN benchmark series were more sophisticated than their predecessors in a number of ways. Especially heteroscedastic linear discriminant analysis, speaker adaptive training and discriminative training were commonly applied in this benchmark event for the first time (see figure 2.8).

For improved feature vectors, most participants used Heteroscedastic Linear Discriminant Analysis (HLDA) [MS03]. HLDA is a method for estimating a linear projection of a vector with a certain dimensionality on a smaller sub-space. Instead of using the first two derivatives of MFCC or PLP features, the first three derivatives are added to the vector. HLDA is then used to find the most important components of the feature and to reduce the dimensionality (mostly back to 39). In the BN benchmark of '98 [BAHU+99] already used Linear Discriminant Analysis (LDA) in order to reduce the vector dimensionality, but in this benchmark series virtually all participants used HLDA [KEH+03, NAA+04, VSW+04].

A technique called Speaker Adaptive Training (SAT [AMSM96]) was commonly applied. Earlier benchmarks had already shown that it helps to normalize features as much as possible (CMN/CVN, VTLN, see section 2.5.1) so that the variety in training data for each phone is minimized and the model parameters can be determined more precisely. SAT uses this same principle of minimizing training set variability. The variation in speech among the speakers of the training set is minimized by modeling the speaker characteristics explicitly as linear transformations of the acoustic parameters during the EM training procedure. This way, the acoustic parameters will be modeled truly speaker independently. Descriptions of systems using SAT can be found in [KEH+03, NAA+04, VSW+04].

Also new in this benchmark event was the use of discriminative acoustic model training methods (for example [KEH+03, VSW+04]). In EM training, for each phone the model parameters are optimized only using its own training data. Discriminative training methods aim at increasing discriminability by also using out-of-class data, the training data for other phones. Because discriminative methods are computationally expensive compared to EM, they weren't used during the earlier BN benchmark tests.

Due to the real-time constraints, it was not possible for the combined system of the two participant groups LIMSI and BBN to merge the results using ROVER [SCD+04]. Instead the results of the ASR run with the LIMSI decoder were used to adapt the model parameters of the BBN decoder. The results using this *cross-system acoustic adaptation* method were equally good as using ROVER. In 2004 when both decoders had become faster, a combination of ROVER and cross-system adaptation was performed [NAA+04].

2.2.5 The meeting rich transcription benchmark series

In contrast to the BN benchmark series, the Rich Transcription (RT) benchmark for meetings only consists of spontaneous speech. The meetings under evaluation typically involve three to eight people discussing various unknown topics. The meetings contain a lot more overlapping speech than the BN recordings and the audio conditions are

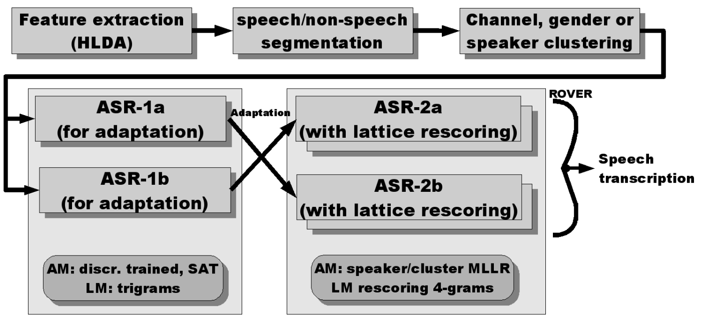

Feature extraction (HLDA)
speech/non-speech segmentation
Channel, gender or speaker clustering
ASR-1a (for adaptation)
ASR-1b (for adaptation)
Adaptation
ASR-2a (with lattice rescoring)
ASR-2b (with lattice rescoring)
ROVER
Speech transcription
AM: discr. trained, SAT LM: trigrams
AM: speaker/cluster MLLR LM rescoring 4-grams

Figure 2.8: *System overview of most systems that participated the 10xRT BN benchmarks in 2003 and 2004. HLDA projection is performed for feature extraction, speaker adaptive training and discriminative training are often used for AM training and sometimes cross system model adaptation is used in the adaptation phase.*

different as well. Although each speaker is equipped with his or her own microphone (close-talking microphone or Individual Headset Microphone (IHM)), the main task is to recognize the speech captured by microphones mounted on the walls or placed on the table. The fact that the characteristics of these Multiple Distant Microphones (MDM) vary per recording and that in general these recordings are of a lower quality than the IHM recordings, makes the MDM task difficult and challenging.

A new task in the RT benchmark series is *speaker diarization*. Participating systems need to determine fully automatically: 'who spoke when?'. Each speaker of the recording needs to be assigned a unique ID and the exact moments that each speaker is talking needs to be annotated[3]. Most systems in the BN benchmark series already contained some form of clustering, but these clustering components were mainly used for adaptation purposes and often did not output the actual number of speakers but just a fixed number of clusters.

Because the audio signal of most MDM microphones are noisy, most participants use some sort of software filter for noise reduction. The multiple signals are processed differently by each site, but combining the signals into one (hopefully) cleaner signal before feeding it into the ASR or diarization system is a popular method. Once the signals are combined, ASR systems similar to the systems described in the previous section are employed. The cross-system adaptation method as applied earlier by [NAA+04] is now often used [HBD+07, WSK07, HMV+07, SAB+07]. Mostly multiple systems are created by using different feature types as input. In [HMV+07] not the feature type, but only the acoustic models themselves where varied. Different models are created by not always choosing the question for the decision tree with the highest increase in likelihood when determining context-dependent clusters (see section 2.1.5), but choosing from the top five options randomly.

[3]Note that speaker diarization should not be confused with speaker identification. Speaker identification is a classification task where should be identified if a recording of someone's voice actually is or is not from that particular speaker.

Because there is only a limited amount of meeting audio available for training ASR models, in [HBD$^+$07] the AMI team used all channels of the MDM recordings separately. Because the characteristics of the microphones vary, they used a speaker adaptive training approach called CHannel Adaptive Training (CHAT). As in SAT training, the channel characteristics were modeled during training and the acoustic models trained without these characteristics were truly channel independent. This proves that the concept of SAT can also be used to create models that are more robust for channel variation.

2.2.6 Existing techniques for creating robust systems

This section overviewed three NIST benchmark series that are highly relevant to the goal of this thesis: to build a Dutch LVCSR system that is robust against unknown audio conditions (see section 1.6.1). A number of techniques that where used in these benchmarks to improve the ASR result were discussed in this section. A selection of these techniques will be investigated further in the following sections.

The systems described in this section all perform segmentation and clustering before running ASR. In the meeting benchmarks, speaker clustering is even a task on its own. Without the segmentation and clustering steps it is not possible to apply the various techniques that make the acoustic models more robust and improve recognition accuracy. In the next two sections existing segmentation and clustering algorithms are presented.

For decoding, a number of techniques to improve feature extraction and acoustic modeling have been discussed. These techniques will be further investigated in section 2.5.

2.3 Segmentation

In this thesis segmentation is defined as the process of cutting up an audio stream in segments *and* labeling these segments with a specific class such as 'silence', 'speech' or 'music'. The main reasons to perform segmentation in a LVCSR system is to filter out the parts of the audio that the decoder won't be able to handle and also to provide the decoder with extra information about the segments so that decoding can be optimized. A third reason to perform segmentation is to enrich the ASR output with the segment information.

The earlier mentioned silence and music classes are obvious examples of audio classes that the decoder won't be able to process successfully. Discarding these segments will speed up the decoding process and most likely keep the word error rate low (fewer insertions). Classes that are often used directly to optimize the decoding process are the 'audio channel' (telephone/broadband) and 'gender' (female/male) classes. For example, the system can train gender specific acoustic models and during decoding use each model according to the segmentation information.

A more complex, but also effective way of optimizing the decoding process is to cluster all segments of the same class together and use this larger amount of audio for unsupervised adaptation of the acoustic models or feature vectors. Adaptation can often be done with higher precision when more data is available and therefore it is interesting to group segments with the same acoustic characteristics together. For example, the first step in finding all segments containing only speech from one single speaker is called *speaker segmentation* or *speaker change detection*. In this case, segments labeled with speech are split at the points in time where a speaker change is detected.

Three main approaches of performing segmentation can be defined: silence-based, model-based and metric-based [CG98, KSWW00]. In the remainder of this section these three approaches will be described briefly. In the next section the process of clustering segments is discussed.

2.3.1 Feature extraction for segmentation

As segmentation is a statistical classification problem, just as for decoding a set of acoustic observations is needed. Although feature types such as MFCC or PLP were not designed to distinguish between speakers, most state-of-the-art segmentation systems actually use these feature extraction methods. For speaker change detection, sometimes feature vectors with a higher number of coefficients are used. A nice example of a system that does not use standard MFCC or PLP features is [AMB03]. Here ‘entropy’ and ‘dynamism’ are used to classify between speech and music.

2.3.2 Silence-based segmentation

For some tasks it is assumed that the audio only contains speech and silence. For example, BN recordings might contain some jingles, but the major part of the recording consists of speech and small pauses between utterances or topics [HOvH01]. Some systems make use of this by segmenting on basis of the silences in the audio. If the segments are later needed to cluster speakers, these systems assume that there is always a short silence between speakers. In case of BN recordings, this assumption is often valid. Unfortunately, for recordings with more spontaneous speech such as recordings of meetings, this assumption is often not valid at all.

There are two common methods of finding silences in an audio stream. The first method is calculating the energy of short (often overlapping) windows. The local minima of this energy series are considered silence. The second method, *decoder-based segmentation*, is to run a fast ASR decoder [WSK07]. Most decoders contain a silence ‘phone’ that takes care of pauses between speech.

In [PH03] the ASR acoustic models are used to create two special models: one for silence and one for speech. The speech model is created by combining the most dominant Gaussian mixtures of all phones into one GMM. A small HMM is then created containing only two states. The first state uses the silence GMM for its PDF and the second state uses the speech GMM. A Viterbi decoding run using this HMM will result in the speech/silence segmentation. Decoder-based segmentation systems, although they only distinguish between silence and speech, can also be considered to be model-based segmentation systems.

2.3.3 Model-based segmentation

Model-based segmentation systems train one GMM for each segmentation class. These GMMs are used as PDF in a hidden Markov model where each state is connected to all other states. Performing a Viterbi decoding run using this HMM results in the segmentation of an audio file. The advantage of this method is that it is very easy to add segmentation classes. The systems in [HJT+98, GLAJ99] train a silence, speech and music GMM, but it is possible to create models for other classes such as sound effects or even known speakers (for example the anchor-man in BN recordings).

Without taking special measures, HMMs with one state for each class tend to produce short segments, even when the transition probabilities from one class to the other are set low. In order to force minimum time constraints on segments, sometimes HMMs are created with a string of states per class that each share the same GMM. Each state in a string is connected to the next state and only the final state has a self-transition (see figure 2.9). The number of states in the string determine the minimum time of each segment. Another approach is to post-process the segmentation and join short speech segments or remove short silence segments.

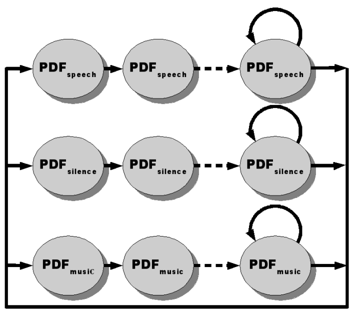


Figure 2.9: *An example HMM used in model-based segmentation. Each string of states represents one segmentation class and all states of a string share the same PDF.*

The major disadvantage of model-based segmentation is that the GMMs need to be trained on some training set. If the acoustic characteristics of the audio under evaluation are too different from the characteristics of the training data, the accuracy of the segmentation will be poor. Model-based segmentation has recently been used in various systems for finding speech and non-speech regions [HJT+98, GLAJ99, HMV+07, SAB+07, vLK07].

2.3.4 Metric-based segmentation

One of the most common segmentation methods to date is *metric-based* segmentation. In metric-based segmentation, a sliding window is used to investigate a short portion of the audio at each step. Typically, the window is cut in the middle and it is determined if this point in time should or should not be marked as a segment border. Some kind of distance metric is used to measure whether the two segments S_i and S_j belong to the same class S, or if they are actually part of two separate segments.

In the literature, a number of distance metrics have been proposed. Most of these metrics make use of models (often Gaussians or Gaussian mixtures) that are trained on S_i, S_j and S in order to calculate distances [Ang06]. The most common distance metric is the *Bayesian Information Criterion* (BIC) [Sch78]. This metric uses some model M_i with $\#(M_i)$ parameters representing a segment of data S_i with N_i time frames (feature vectors) and it determines how well the model fits the data:

$$\mathrm{BIC}(M_i) = \log L(S_i, M_i) - \frac{1}{2}\lambda\#(M_i)\log N_i \tag{2.6}$$

λ is a free parameter that needs to be tuned on a training set. The value of this parameter influences when the BIC value is positive, meaning that the model fits the data, or negative, meaning that the model does not fit the data very well. Formula 2.6 can be used to determine if the data of the two segments S_i and S_j fit M_i and M_j best or if the data of the two segments together (S_i+S_j=S) fit the model M trained on S the best:

$$\begin{aligned}\Delta\mathrm{BIC}(M_i, M_j) &= \mathrm{BIC}(M) - (\mathrm{BIC}(M_i) + \mathrm{BIC}(M_j)) \\ &= \log L(S, M) - (\log L(S_i, M_i) + \log L(S_j, M_j)) \\ &\quad - \lambda\Delta\#(M_i, M_j)\log N\end{aligned} \tag{2.7}$$

where $\Delta\#(M_i, M_j)$ is $\#(M) - (\#(M_i) + \#(M_j))$. If ΔBIC is negative, the model of the total segment S fits the data not as good as the two separate models and a segment border is placed between the two segments. ΔBIC was first used for segmentation and clustering in [CG98]. In [Ang06] a mathematical proof of formula 2.7 is given. Note that when $\Delta\#(M_i, M_j)$ is zero, meaning that the number of free parameters in M equals the number of free parameters in M_i and M_j, the design parameter λ no longer influences the equation.

In combination with speaker clustering, the Bayesian Information Criterion has recently been used for speaker change detection in a number of systems [Cas04, IFM$^+$06, vLK07, RSB$^+$07].

2.3.5 Assessment of segmentation systems

Two measures are regularly used for assessing segmentation results. The first one is to measure for each class the percentage of time that the class was correctly assigned, or if one overall number is required, the percentage of time that all classes were correctly assigned:

$$\mathrm{Score} = \frac{\sum\limits_{c}(C_c)}{L} \cdot 100\% \tag{2.8}$$

where C_c is the total time that class c was classified correctly and L is the total audio length.

The second method of assessing segmentation systems is to consider the result to be a special case of a speaker diarization system [NIS06]. This was done at the NIST benchmarks in 2005 and 2006 for the Speech Activity Detection (SAD) task. SAD is a segmentation task with two classes: speech and non-speech. All reference speakers were joined in one cluster and any speaker overlap was removed. The measurement explained in section 2.4.2 was then used to score the SAD results. Because overlapping speech is not measured and therefore the number of 'speakers' is always zero or one, for SAD systems, formula 2.10 in section 2.4.2 can be formulated as:

$$\text{SAD} = \frac{M + F}{S} \cdot 100\% \tag{2.9}$$

where S is the total time of speech, M is the total time of speech that was not classified as speech (missed speech) and F is the total time of silence that was falsely classified as speech (false alarms). Note that this measurement results in an error percentage while the first measurement (formula 2.8) results in a percentage of correctly assigned classes. Also, the SAD measurement is a percentage of the total time of *speech* in the reference transcript, while using the first measurement for a SAD system would result in a percentage of the total time of the evaluation audio.

2.4 Clustering and speaker diarization

Clustering acoustically similar speech segments is helpful for adaptation of the acoustic models or feature vectors. For most adaptation techniques the individual segments are too short to accurately determine the values of the parameters needed for adaptation (see section 2.5.2) and therefore the segments that are most similar are clustered and the entire cluster is used for adaptation.

The most common technique used for clustering is *hierarchical clustering*. In fact, all ASR teams of the RT07s benchmark [HBD+07, WSK07, HMV+07, SAB+07, LBG+07] applied this clustering technique. Hierarchical clustering can be done in two ways: either *top-down* or *bottom-up*. In top-down clustering, initially all speech segments are placed in one cluster and iteratively the cluster is split into multiple clusters until an optimum number of clusters is reached. In bottom-up clustering, also called *agglomerative clustering*, a high number of initial clusters is initially created (sometimes even one speech segment per cluster) and the clusters are iteratively merged until an optimum number of clusters is reached.

For ASR purposes, the optimum number of clusters is not necessarily the exact number of speakers in the audio recording. Because it is important that a minimum amount of data is available for acoustic adaptation, speakers that did not speak long enough can be placed in a cluster with other speakers (as done for example by [SAB+07]). For the *speaker diarization* task of the NIST rich transcription benchmark series though, where the task is to determine automatically 'who spoke when?', obviously it is the goal to assign exactly one cluster to each speaker of the recording.

This requirement led to the use of a number of new, more accurate, approaches for clustering. In this section, first the agglomerative clustering method is described and then a number of systems that participated in the NIST speaker diarization benchmarks is discussed.

2.4.1 Agglomerative clustering

Agglomerative clustering consists of four iterative steps. First, initial clusters need to be defined. Often, each speech segment that is found during segmentation is considered a single cluster. Next, the distance between these clusters needs to be determined. Often this is done pairwise and the distance between each pair of clusters is stored in a matrix. This matrix is used in the third step to determine if there are any clusters that can be merged into one cluster or if the optimum number of clusters is reached. If this *stopping criterion* decides that the optimum is not yet reached, in the fourth step the clusters with the smallest distance are merged and the process is iterated starting at the second step. The distance metrics that are used to determine the distance matrix, are often the same metrics as used during segmentation. This means that for each cluster a model is created and during the merging phase a new model is created for each pair of clusters that are merged.

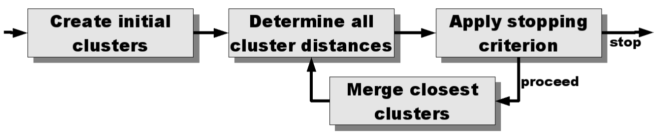


Figure 2.10: *Agglomerative clustering is an iterative process where a high number of initial clusters is iteratively reduced to the optimum number of clusters.*

A popular metric used for clustering is BIC (see section 2.3.4). Similar to the usage of BIC during segmentation, for each cluster and for each pairwise combination of clusters a model is created. For segmentation purposes, only a model trained on data of two segments needs to be trained when those segments are bordering each other, but when BIC is used for clustering, the BIC score for each combination of clusters is calculated and a combined model of each combination of clusters is needed. When segmenting, the segment data can be modeled by a single Gaussian (with full covariance matrix), but one single Gaussian is not able to model the clusters with enough precision. Therefore, GMMs are mostly used. The λ parameter (equation 2.7) needs to be tuned on a training set, but when the number of Gaussians (the number of free parameters) in the combined model equals the sum of the number of Gaussians of the cluster models, the λ factor is eliminated (section 2.3.4). The higher the value of ΔBIC, the more similar the two clusters that are being compared. Therefore, the clusters with the highest ΔBIC score will be considered for merging first.

Not only the distance matrix can be calculated using BIC. Often, BIC is also used as stopping criterion. When the ΔBIC score is negative, the two separate models represent the clusters better than the combined model and therefore the clusters should not be merged. In other words, when all values of the distance matrix are negative, the process should stop. Otherwise, the clusters with the highest scores can be merged. As mentioned earlier, for some applications BIC is not used as stopping criterion, but merging is repeated until a fixed number of clusters is reached.

2.4.2 Assessment of speaker diarization systems

Speaker diarization systems need to segment *and* cluster audio recordings on speakers. The metric used to evaluate the performance of these systems is called Diarization Error Rate (DER) [NIS07]. It is computed by first finding an optimal one-to-one mapping of the reference speaker segments to system output and then obtaining the error as the fraction of time that the system did not attribute correctly to a speaker or to non-speech. Finding the optimal mapping is needed because the system does not need to identify speakers by name and therefore its speaker labels will differ from the labels in the reference transcript. The DER is calculated as follows:

$$DER = \frac{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot (max(N_{\mathrm{ref}}(s), N_{\mathrm{sys}}(s)) - N_{\mathrm{correct}}(s))}{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot N_{\mathrm{ref}}} \tag{2.10}$$

where S is the entire set of segments. In this case, a segment is defined as a fragment in the audio in which no speaker change is occurring in either the reference transcription or the hypothesis transcription. Because of overlapping speech it is possible that multiple reference or hypothesis speakers are talking during one segment, but the number of reference speakers $N_{\mathrm{ref}}(s)$ and the number of hypothesis speakers $N_{\mathrm{hyp}}(s)$ do not change during one segment s. Further, $\mathrm{dur}(s)$ is the duration in seconds of segment s and $N_{\mathrm{correct}}(s)$ is the number of reference speakers speaking in s for whom their mapped hypothesis speakers are also speaking in s.

Another metric, mainly used for system analysis, is hypothesis speaker *purity*. Each speaker in the hypothesis SPK_h is mapped to the reference speaker SPK_r that is represented the most amount of time by that hypothesis speaker. The purity of a hypothesis speaker is then:

$$Purity(\mathrm{spk}_h, \mathrm{spk}_r) = \frac{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot \mathrm{SPK}(s, \mathrm{spk}_h) \cdot \mathrm{SPK}(s, \mathrm{spk}_r))}{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot \mathrm{SPK}(s, spk_h)} \tag{2.11}$$

where $\mathrm{SPK}(s, spkr)$ is one when speaker *spkr* is talking in segment s and zero otherwise. Note that it is possible that multiple hypothesis speakers are mapped to the same reference speaker. If the purity of a hypothesis speaker is a hundred percent, this means that all speech of this speaker can be mapped to one single reference speaker and no noise of other speakers is present. This does not mean though that the hypothesis speaker will not contribute to the DER as the reference speaker might be mapped to another hypothesis speaker. In chapter 5 purity is used for analyzing a diarization system.

2.4.3 NIST benchmark series for speaker diarization

Since the year 2000, NIST has been organizing benchmarks for speaker diarization. At first telephone speech was evaluated (2000, 2001 and 2002), later broadcast news (2002, 2003 and 2004), and now evaluation focuses on the meeting domain (2002, 2004, 2005 and 2006). Since 2004, the speaker diarization task is part of the rich transcription benchmarks. The same data is used for diarization as for ASR, only the close talking microphones are not used. This leaves two main conditions for diarization: the single distant microphone and the multiple distant microphone recordings.

Numerous interesting speaker diarization systems have participated in the NIST rich transcription benchmark series for meetings. Mostly metric-based segmentation and clustering is being used [Cas04, vL06, RSB$^+$07, IFM$^+$06], but also model-based segmentation is popular [ZBLG07, FS07, AWP07]. The Bayesian Information Criterion is used most as distance measure for all three tasks: segmentation, picking models to merge and as stopping criterion [Cas04, vL06, RSB$^+$07]. But also other measures such as the Generalized Likelihood Ration (GLR) [MFP$^+$04, JLSW04, IFM$^+$06] and Mahalanobis distance [Cas04] have been used for segmentation or clustering. A hierarchical top-down clustering approach was taken by [FS07] using an HMM adding new states (representing speakers) at each iteration while re-segmenting the speech data. In [ZBLG07, AWP07] a bottom-up approach was taken, also using an HMM to realign the data after each iteration.

The main condition of the benchmark series was the Multiple Distant Microphones (MDM) condition where for each meeting, multiple recordings were available. Some systems just picked a single channel [vL06, ZBLG07], while others segmented each channel separately [JLSW04] before combining the results or performed some form of pre-processing [FS07, AWP07] to combine the channels into one single recording.

In [Ang06], a good short description of all these systems is provided. Although every one of them is interesting, only one single system, the speaker diarization system of the International Computer Science Institute (ICSI), will be described in depth in this section. The ICSI system is extra interesting because of its consistently high performance at the benchmark series[4] and because of its strategy not to use any models or parameters that need to be tuned on training data [AWPA06, AWP07]. The absence of the need of training data during system development, makes the system robust for unknown audio conditions. No training data means that it is not possible to have a mismatch between training and evaluation data.

2.4.4 The ICSI speaker diarization system

The speaker diarization system of the International Computer Science Institute (ICSI) is based on a system originally described in [ABLM02]. First the ICSI system participated the speaker diarization benchmarks for BN (2003 [AW03] and 2004 [WFPA04]) and later for rich transcription of meetings [AWPA06, AWP07, WH08]. The system consists of three main components: feature extraction, speech activity detection and

[4]The rules of the benchmark series prohibits publication of ranking results, but these rankings can be found at the web page of each benchmark: `http://www.nist.gov/speech`

speaker diarization. These components will be described as they were implemented for the rich transcription benchmarks in 2005 and 2006 [AWPA06, AWP07]. The work performed on this system in 2007 will be described later in this thesis.

This section will provide an overview of the ICSI speaker diarization system. In [Ang06] the system itself and also the various techniques aimed at improving the basic system are discussed in-depth.

Feature Extraction

The meetings under evaluation are recorded with multiple distant microphones. The audio signal of each microphone is first passed through a Wiener filter for noise reduction where noise is assumed to be additive and of a stochastic nature [WN49]. The implementation of the Wiener filtering that was used, was taken from the noise reduction algorithm developed for the Aurora 2 front-end proposed by ICSI, OGI and Qualcomm [ABD$^+$02]. After Wiener filtering, the channels are combined into one 'enhanced' channel using delay and sum beamforming software (BeamformIt[5]). This software determines the delay of each signal relative to the other signals and removes this delay before summing all signals together [Ang06].

From the resulting 16kHz audio file, Mel Frequency Cepstral Coefficients (MFCC) are extracted. The feature vectors are calculated using 30ms Hamming windows that are shifted 10ms at a time. Twenty-four melscale filters are used to calculate vectors containing the first nineteen cepstral coefficients. A second feature stream is created by using the BeamformIt tool to calculate the delay values between the different audio channels. When using BeamformIt to produce these delay features, a 10 ms step size is used instead of the 250 ms step size used for performing beamforming. This way, the second feature stream contains the same number of vectors as the MFCC feature stream.

Speech activity detection

In 2005 speech/non-speech segmentation was performed using a model-based approach [AWPA06]. One model for speech and one model for non-speech were trained on a training set of meetings. Each model contained three states and the first twelve MFCC coefficients were used as input. The downside of this system was that new models needed to be trained as soon as the evaluation conditions changed. Therefore in 2006, the system was replaced by a two step algorithm [AWP07]. First, a silence-based set-up was used to find all segments with low energy. It was assumed that silence was the only form of non-speech in the meetings and that this first step was able to find enough representative speech and silence segments to use in the second step. In this second step, the segments were used to train a model-based system with two states: one for speech and one for non-speech. The HMM was used to realign the data and using the new alignment, new GMMs were trained. After a number of iterations the final speech/non-speech alignment was obtained.

[5]`http://www.icsi.berkeley.edu/~xanguera/beamformit`

The advantage of the 2006 system is that no training data is needed for the speech and non-speech models. The downside is that this system is only able to distinguish between speech and silence. Because in the first step an energy-based segmentation is created, all non-speech with high energy will be classified as speech. In the second step this error will not be corrected. Fortunately, the assumption that all non-speech in the meetings under evaluation is silence is often valid.

Speaker diarization

The diarization system is based on the use of HMMs with GMMs as probability density functions. The HMM is identical to the topology earlier discussed in figure 2.9. The string of HMM states drawn horizontally each represent one speaker and all states in a string share a single GMM. In an ideal situation, each GMM is trained on all the speech of one unique speaker. The speaker segmentation, the final system result, is found by performing a Viterbi alignment of all audio that contains speech. All audio that is processed by the same string of states during this alignment is grouped together as speech from one speaker. By using a string of states to represent each speaker (instead of a single state), a minimum duration of each speech segment is guaranteed.

Clustering is done using the agglomerative clustering method. Initially too many HMM states are created. The number of states is then iteratively decreased and the GMMs are slowly trained on speech from a single speaker until the correct number of GMMs is reached. In order to obtain this optimal topology, an algorithm with five steps is executed. An overview of these steps is drawn in figure 2.11.

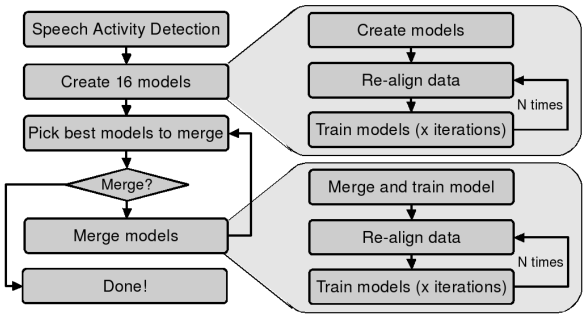


Figure 2.11: *A schematic representation of the speaker diarization algorithm. In order to evaluate the five steps of this algorithm, each step can be replaced by an oracle component. The second and fifth step each consist of a number of training and re-alignment iterations.*

First, SAD is performed as described earlier. The speech segments are then passed to the speaker diarization system for initialization.

The system is initialized with a large number of models (strings of HMM states). The number of models should be significantly higher than the assumed maximum number of speakers in the audio file. For the meeting benchmark series, 16 initial clusters for each meeting were used. In order to create the initial sixteen speaker models, the available speech data is cut up into small pieces and these pieces are randomly divided in a number of bins. Each bin is used to train one of the multi-stream (MFCC and delay-sum features) GMMs with five Gaussians. Although the GMMs are trained using multiple speakers, in general one speaker will fit the GMM a little bit better than the other speakers. Therefore, when a Viterbi alignment is performed, this GMM will be assigned more speech from this speaker. The data will be re-aligned a number of times and after each iteration, the GMMs are re-trained. After each iteration the model will fit the dominant speaker better than before. The result of this step is a group of models that are all trained with as much data as possible of one dominant speaker and as little data as possible of the other speakers. In the remaining steps, the models that are trained on the same dominant speaker will need to be merged.

In the third step it is determined which two models are most likely trained on the same speaker. This is done by calculating the local ΔBIC score for each combination of two models (see formula 2.6). For this BIC comparison, a new model is trained containing the sum of the number of Gaussians of the two original models. This *merged* model is trained on the training data of the original two models. As discussed in section 2.3.4, no scaling parameter is needed in the ΔBIC equation, because the total number of Gaussians, the number of free parameters, will remain unchanged after replacing the two models by the merged model. If the BIC score is positive, the two original models are considered to be trained on data of the same speaker. The higher the BIC score, the more the two models were similar. Therefore, the cluster pair with the highest BIC score will be chosen as the candidate for merging and in the fourth step, the decision to merge the candidate cluster pair is positive if the BIC score is bigger than zero and negative if the BIC score is negative.

In this last case the final Viterbi alignment will be performed and the algorithm is finished. But if the BIC score is positive, in the fifth step, the two merge candidates are replaced in the HMM by the merged model. The data is re-aligned over the models and the models are re-trained with the new data. After this, a new merging iteration is started.

System parameters

The speaker diarization system is designed to have no system parameters that need to be tuned on external data, making it robust for changes in audio conditions or application domain. Because the GMMs are trained on the data under evaluation, no models created on a training set are needed. Also, because BIC is used with preservation of system complexity, no scaling parameter is needed. Unfortunately, five system parameters still exist, although these parameters do not seem sensitive for changes in audio conditions [AW03]. The first parameter is the number of initial clusters (set to 16), the second parameter is the number of Gaussians of each initial GMM (set to 5). Third, the number of states in each string, the minimum segment duration is a tunable parameter (set to 250, 2.5 seconds). The final two parameters are the number of training iterations used for GMM training and the number of times that the data is re-aligned during each training run.

The first ICSI diarization system that used the sum delay information as a second feature stream, proposed in the meeting benchmark of 2005, needed to tune the stream weights on a training set [AWPA06]. In 2006 though, a method of automatically tuning these weights during evaluation was proposed [AWP07] and the training set was no longer needed.

2.5 Techniques for robust ASR

In section 2.2 a number of techniques were introduced that is commonly used to improve robustness in ASR. Some of these techniques are designed to *normalize* the features and acoustic models so that any unwanted differences in data is minimized. The normalization procedure is used during training to minimize differences in training data which makes it easier to create appropriate models, and it is also used during decoding so that the evaluation data fit the models as good as possible.

Other techniques aim at *adapting* acoustic models so that they better fit their task. For example, acoustic models can be adapted for optimally decoding a specific speaker or specific recording conditions. For adaptation of acoustic models, *adaptation data*, consisting of time aligned occurrences of the phones, are needed that can either be obtained by manually time aligning the data (supervised adaptation) or by using the results of the decoding run (unsupervised adaptation). In the case of unsupervised adaptation two decoding runs are needed. The results of the first run are used to adapt the models and these models are applied in the second decoding run.

Next, in section 2.5.1 techniques that are based on normalization are discussed. The acoustic model adaptation techniques are discussed in section 2.5.2.

2.5.1 Feature and acoustic model normalization

Countless techniques for feature and AM normalization have been proposed in the literature. Here, a small subset of these techniques that was repeatably used during the NIST benchmarks will be described.

Audio preprocessing

In the RT meeting benchmarks, the meetings under evaluation are recorded with multiple distant microphones. Because farfield microphones in a meeting setting are more likely to pick up noise than the microphones in broadcast news studios, reducing noise is an important preprocessing step for systems in the RT meeting benchmarks. A common method of reducing noise is to pass the audio signal of each microphone through a Wiener filter [WN49]. With Wiener filtering the noise is assumed to be additive and of a stochastic nature. An implementation of Wiener filtering that is used by multiple teams, is taken from the Aurora 2 front-end proposed by ICSI, OGI and Qualcomm [ABD+02].

When multiple microphones are used for recording, it is possible to create a single recording that is in general of higher quality than the separate signals. This 'enhanced' channel is created using the delay and sum beamforming method. This method determines the delay of each signal relative to the other signals and removes this delay before summing all signals together [Ang06]. The resulting single and noise reduced signal can then be used to perform feature extraction.

Cepstrum mean and variance normalization

Cepstrum Mean Normalization (CMN) and Cepstrum Variance Normalization (CVN) are well known normalization techniques that are applied to the individual cepstrum coefficients of the feature vectors. When applying CMN, the mean of each coefficient is normalized to zero and with CVN, the variance of each coefficient is normalized to one. For this, first the actual mean and variance are calculated and the mean is then subtracted from each coefficient and the result is divided by the variance. CMN and CVN can be conducted on an entire recording or on parts of the recording (such as speech segments or speaker clusters). For online ASR applications it is also common practice to perform CMN and CVN on a running window.

Vocal tract length normalization

Variation of vocal tract length between speakers makes it harder to train robust acoustic models. Vocal Tract Length Normalization (VTLN) can be applied to normalize for this variation. Using VTLN, the Mel-scale windows of the feature vectors are shifted by a certain warping factor. If the warping factor is smaller than one, the windows will be stretched and if the factor is bigger than one, the windows will be compressed. To normalize for the vocal tract length, large warping factors should be applied for people with a low voice and small warping factors for people with a high voice[6]. For the best performance, VTLN is not only used during decoding, but also during training of the acoustic models.

Typically, the warping factor is determined by performing a grid search. A range of warping factors is used during the decoding of a number of utterances of a speaker. The warping factor with which the hypothesis with the highest score is obtained is then chosen as the warping factor for that particular speaker. This procedure requires a number of decoding runs which is computationally expensive. In order to speed up the process it is possible to decode speech with an AM that was not normalized and then use a range of warping factors to perform forced alignment of the hypothesis on the normalized features. Again, the warping factor is chosen that was responsible for the forced alignment with the highest score. Unfortunately, this approach still requires an initial decoding pass.

In [WMOP96] a fast method to determine warping factors was proposed. Instead of decoding a number of utterances, all speech features, normalized using a range of warping factors, are passed through a single GMM that was trained on all speech from a collection with balanced male and female speakers. The warping factor that reproduces the highest score will be picked as warping factor for the particular speaker.

[6]note that according to how the factor is used for shifting the windows, sometimes this is reversed: sometimes big warping factors are linked to high voices instead of low voices

In order to obtain the most accurate results, the general speech GMM is created in a number of iterations. After each iteration, the warping factors of the training speakers are determined and a new model is trained using features that are normalized with the newly obtained warping factors.

2.5.2 Acoustic model adaptation

The goal of acoustic model adaptation is to change the acoustic model parameters, say θ, in a way so that they will perform better when decoding specific data. For example, speaker independent models may be adapted to perform well for a specific speaker (and become a speaker dependent model) or for a specific noisy recording condition. In order to adapt the AM parameters, example data $Y = \{y_1, y_2, ..., y_T\}$ of a specific speaker or recording condition, the *adaptation data*, are needed. The two most common methods for acoustic model adaptation are Maximum Likelihood Linear Regression (MLLR) and Maximum a Posteriori (MAP) adaptation. In most systems these methods are used to adapt the Gaussian means of the AMs. Sometimes, also the Gaussian variance is adapted.

Maximizing the likelihood or the posterior

With Maximum Likelihood Linear Regression (MLLR), the model's Gaussian means or variances are adapted using a linear transformation [LW95]. For this, similar to training the models, the Expectation Maximization (EM) theorem is applied. EM is used to optimize θ by iteratively optimizing the *likelihood* of the models on the adaptation data (formula 2.12).

$$\text{MLLR} : \hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(Y \mid \theta) \tag{2.12}$$

Instead of maximizing the likelihood, Maximum a Posteriori (MAP) adaptation maximizes the posterior [LLJ90]. Because this is done in the Bayesian framework (see formula 2.1), MAP is also called Bayesian adaptation. In order to maximize the posterior, not only the likelihood, but also a meaningful *prior*, $P(\theta)$, of the parameter settings is needed. This prior is not the same as the prior for decoding (the language model). Instead the prior should describe the expected optimal values of the parameter settings θ (see 2.13).

$$\text{MAP} : \hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(Y \mid \theta) \cdot P(\theta) \tag{2.13}$$

The prior makes it possible to *restrict* the new parameter settings $\hat{\theta}$ to reasonable expected values. A good prior aids in reducing the risk of over-fitting the model parameters to the adaptation data. Therefore, especially when only little adaptation data is available, MAP outperforms MLLR adaptation. On the other hand, when enough adaptation data is available, in general the prior will restrict the adaptation and MLLR will outperform MAP. Note that when the prior is not *informative*, when it does not actually add information about preferable parameter settings to the framework, MAP is identical to MLLR adaptation.

Regression classes

When sufficient adaptation data is available, it is possible to adapt each model *directly* on its own part of the data, the pronunciations of that specific triphone. When less data is available, it is possible to cluster together data for a number of models and perform adaptation on each cluster. All models from one cluster will then be adapted using one single transformation. This type of adaptation is called *indirect* adaptation and the clusters are generally called *regression classes*. Indirect adaptation makes it possible to adapt models even if little or no adaptation data is available for certain phones.

Regression classes can be chosen manually on basis of knowledge of a language, or based on statistics. For example, in [YEH$^+$95] the Gaussian mixtures of all models are placed in the root of a binary tree and the node is then split on basis of the overall mean of the mixtures. This process is repeated until small regression classes are formed in the leaf nodes.

The choice of regression classes and adaptation method influences the quality of the AM adaptation considerably. In both cases, the optimal choice is highly dependent on the available adaptation data. Regression classes need to be optimized for the available data as too general classes are likely to result in no or little adaptation and small classes might result in over-fitted models. Similarly, MLLR is best used for small amounts of data, while MAP performs best when more data is available. In [SML00] a method called Structured Maximum a Posteriori Linear Regression (SMAPLR) is introduced that is less sensitive for these choices and is therefore more robust when applied for unsupervised adaptation where it is not beforehand clear how much adaptation data will be available.

Structured Maximum a Posteriori Linear Regression

In [SML00], Structured Maximum a Posteriori Linear Regression (SMAPLR) is introduced. This adaptation method consists of two steps. First, a tree is created that contains one regression class in each leaf node. Second, MAP adaptation is performed on each regression class. The robustness of this method lies in the method chosen for creating the regression classes and in the method of defining the prior for each regression class.

For creating the tree, the method in [SL98] was used where the tree is created by clustering bottom-up. Small regression classes are created and clustered using Kullbach divergence until one single cluster is formed in the root node of the tree. The initial classes can for example be triphone GMMs or perhaps all triphone models of each phone grouped together. Next, in order to avoid regression classes that are represented by too little data, a minimum amount of data that should be available for each class is enforced. If a leaf node is represented by less than the minimum amount of data, it is removed from the tree and its parent node automatically becomes the leaf node.

Starting in the root node, MAP adaptation is performed on all nodes in the tree. The posterior probability obtained at each node is passed to its children nodes. The children nodes use the posterior probability of their parent as prior probability for their own MAP adaptation. For the root node, the prior is not informative and therefore at the root node, MLLR adaptation is performed. It is assumed that the transformation obtained in parent nodes provide useful information for constraining the child nodes and that at each step deeper into the tree, the priors get more refined as the posteriors are better modeled. When a lot of training data is available, this means that the adaptation transformations can be determined with higher accuracy and at the same time, when not that much data is available, the priors will restrict MAP from over-fitting the data. This means that the system is less sensitive for the minimum needed amount of data that was picked for creating the regression classes [SML00]. That this is actually the case can be illustrated by a simplified example. Let's say that all probabilities are represented by single Gaussian functions that are modeled based on the adaptation data. When there is a lot of data available in the root of the tree, the posterior of the root node will be a Gaussian with a relatively high variance. Such a Gaussian will not be very restrictive when used as prior and therefore the child node will be able to adapt freely on its own data. On the other hand, when there are less data available in the root node, the variance of the Gaussian will be relatively small. This small variance will act very restrictive when used as a prior in the child node.

2.5.3 Speaker adaptive training

Speaker Adaptive Training (SAT) is a mix of model normalization and adaptation. For each speaker in the training set the acoustic models are first adapted. These speaker dependent models are then used during training of new acoustic models instead of the old speaker independent AM. The transformation matrices used during adaptation can be regarded as information about the speaker that is filtered out before training the new models. This means that the new models will be truly speaker independent. This method can be compared to VTLN where the vocal tract length of each speaker is filtered out before the acoustic models are trained.

The SAT acoustic models do not outperform unnormalized acoustic models in the first decoding run. They do increase decoding precision though when they are used as initial models for speaker adaptation. Therefore, often unnormalized models are used in a first decoding iteration to obtain an initial hypothesis and the SAT acoustic models are then used to create speaker dependent models that are applied in the second decoding iteration [AMSM96].

2.6 Final remarks

In this chapter, the state-of-the-art in speech activity detection, speaker clustering and automatic speech recognition was given. Fundamental techniques such as Viterbi decoding and BIC comparisons were discussed as well as the performance of various systems in the major benchmarks.

The theory described in this chapter is the foundation for the work that will be discussed in the remainder of this thesis. The agglomerative model based clustering method described in section 2.4.4 was used as a starting point for the diarization subsystem introduced in chapter 5 and it was also an inspiration for creating the speech activity detection subsystem described in chapter 4. The ASR subsystem described in chapter 6 makes use of the fundamental ASR techniques discussed in this chapter and incorporates a number of techniques found in the systems discussed in the benchmark section (section 2.2). Before describing the three subsystems in-depth, in the following chapter an overview of the proposed system, called SHoUT will be provided.

CHAPTER 3

THE SHOUT SYSTEM

In this chapter, the ASR system that has been developed to be robust for new, unknown audio conditions will be introduced. The development strategy for this software framework will be discussed in section 3.1. The strategy has been guiding the decisions on which existing techniques to incorporate and on which new algorithms to develop. After discussing the development strategy, in order to avoid confusion, in section 3.2 some system related terms will be defined that will be used in this thesis. In the remainder of this chapter the proposed system and the encountered research topics are described. These research topics are addressed in-depth in the following chapters.

3.1 Development strategy: the fewer parameters the better

Instead of using existing available software as a starting point, a completely new system for automatic speech recognition has been created. An important reason for doing this is that by starting from scratch, no existing software constraints can prevent using new techniques. Of course the newly developed software will have its own limitations, but the system can be developed so that there are no problems incorporating the currently state-of-the-art techniques as described in the previous chapter. Also, the software can be developed so that it is possible and easy to re-use parts of the ASR software for the SAD and clustering subsystems and so that it is flexible for performing research on specific topics (see for example the implementation for Language Model Lookahead in chapter 6). Positive side effects of creating the system from scratch are that there are no legal restrains on the resulting software and that developing the toolkit itself is a valuable learning experience. The applications that were written for this research are grouped together in a software toolkit that will be referred to as the SHoUT toolkit. SHoUT is an acronym for SpeecH recognition University of Twente (or in Dutch: Spraak Herkennings Onderzoek Universiteit Twente). SHoUT is available under the GNU General Public License, version 2.

In section 3.3 the SHoUT system will be introduced. The system consists of three subsystems: speech/non-speech segmentation, clustering and ASR. Segmentation is needed to identify the fragments in the audio that actually contain speech. Speaker clustering is incorporated to the system to make unsupervised adaptation possible on a speaker basis. The ASR subsystem performs the actual decoding.

The strategy is to use as few parameters that need tuning on a training set as possible. Tuning parameters is only useful when the conditions of the training set represent the conditions of the test set. As it is unknown what the conditions of the test set will be, it is hard to tune parameters correctly. It is not needed to create a system that is completely free of parameters. Parameters that are not affected by changing audio conditions, but that are set once to an optimal value can be used without risking a decrease of system performance.

The problem of test set conditions not matching the training set conditions exists for statistical models as well. As the conditions of the training set might not represent the test set, the models that are trained on this training set might perform poor on the test set. Therefore, the use of models created on training data is limited as much as possible.

This approach will prove to be feasible for SAD and clustering, but unfortunately tunable parameters and acoustic models are hard to get rid of for automatic speech recognition. Therefore for ASR, instead of not using models developed on a training set, it is tried to make the models as independent of the audio conditions as possible. A huge amount of techniques are described in literature for normalizing the models (see chapter 2), but unfortunately it is not practically possible to combine all of these approaches in one framework during this research. Therefore a number of existing techniques that proved itself during benchmark evaluations will be used to make the models and also the feature vectors more robust against changing audio conditions. Once an initial speech transcription is generated, it is possible to apply unsupervised adaptation of the models. For acoustic model adaptation the same strategy as before is followed: the fewer tunable parameters the better.

Before an overview of the proposed system is given, some definitions will be formulated to avoid confusion in the remainder of this thesis. The specific research topics defined in the following sections will be discussed in-depth in the following chapters.

3.2 Software architecture, some definitions

The terms used in this thesis to discuss the software architecture of the SHoUT system are based on the definitions formulated in [BCK03]. In this book on software architecture, three types of architectural representations, so called structures, are defined: component-and-connector structures, module structures and allocation structures. The component-and-connector structure defines the functionality of each part of the software (components) and how these parts interact with each other (connections). The module structure defines how the source code, the actual implementation, is structured. The allocation structure defines how the software elements will allocate resources such as processors, files or memory. The discussions in this thesis will focus on the first two types of structures.

For the component-and-connector representation of the software, the following definitions will be used:

- The total of all software building blocks that work together to create speech transcripts from audio recordings is called the *system*.
- The SHoUT system consists of three *subsystems* that each perform a particular task: segmentation, diarization and ASR.
- The part of a subsystem that is responsible for a high level functional procedure is called a *component*. For example, the part of the diarization subsystem that handles picking GMMs to merge is called a component. At the highest abstraction level, each subsystem can be broken up into a number of these functional components.

Although segmentation, diarization and ASR are only steps in the total system and are therefore called subsystems, these subsystems may be used stand-alone. This has proven to be valuable for example in the NIST benchmarks where diarization is not applied as part of a bigger system. In such a context it would be confusing to refer to diarization as a subsystem because this implies that not the entire procedure is described but only a part of it. Therefore, when subsystems are employed for stand-alone tasks, they are not referred to as subsystem, but simply as a system on its own.

For discussions on the module structure of the software architecture, the following terminology will be used:

- The implementation of each component, the actual source code that performs the task defined by the component, is called a software *application*. An application is a single stand-alone software program that does not need any other applications to perform its task.
- A *module* is a collection of source code from an application that is responsible for basic tasks such as GMM or LM handling. Each module is supposed to be reusable for other applications and easily replaceable by alternative module implementations.
- The collection of all applications in the system, but also the helper applications that create the statistical models, is called the *SHoUT toolkit*.

3.3 System description

Figure 3.1 is a graphical representation of the system work flow that starts with speech activity detection (SAD) in order to filter out the audio parts that do not contain speech. In contrast to the SAD subsystem in the BN system, this SAD subsystem needs to be able to filter out all kinds of sounds such as music, sound effects or background noise with high volume (traffic, cheering audience, etc). As a speech decoder will always try to map a sound segment to a sequence of words, processing audible non-speech portions of the data would introduce noise in the transcripts due to assigning word labels to non-speech fragments. Even processing silence fragments is unwanted as the ASR subsystem will claim unnecessary processor time. Also the performance of the clustering subsystem will decrease when its data is polluted with non-speech fragments.

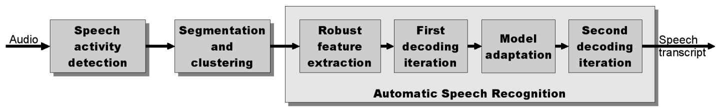

Audio
Speech activity detection
Segmentation and clustering
Robust feature extraction
First decoding iteration
Model adaptation
Second decoding iteration
Speech transcript
Automatic Speech Recognition

Figure 3.1: *Overview of the decoding system. Each step provides input for the following step. There are three subsystems: segmentation, diarization and ASR.*

After SAD, the speech fragments are segmented and clustered. In this step, the speech fragments are split into segments that only contain speech from one single speaker with constant audio conditions (speech from a telephone recording and high quality recordings will be separated, even if the speech comes from only one speaker). Each segment is labeled with its corresponding speaker ID. Next, as part of the ASR subsystem, for each segment feature extraction is performed. In order to create as robust features as possible, the cluster information is used to normalize these features. Decoding is done using an HMM-based Viterbi decoder. In the first decoding iteration, triphone acoustic models and trigram language models are used. For each speaker, a first best hypothesis aligned on a phone basis is created for un-supervised acoustic model adaptation. The second decoding iteration uses these speaker adapted acoustic models to create the final first best hypothesis aligned on a word basis. Also, for each segment, a word lattice is created that can be used for re-scoring.

3.3.1 Speech activity detection

As mentioned earlier, although generally HMM-based SAD systems perform very well in controlled audio conditions, when the training data do not match the evaluation data, the performance can drop significantly. Because of the large variety in collections such as the TRECVID$_{07}$ collection (see appendix A) it is difficult to determine a good set of data on which to train the silence and speech models. Also it is difficult to determine what kind of extra models are needed to filter out unknown audio fragments such as music or sound effects and to collect training data for those models.

The SAD method proposed by [AWP07] at RT06s (chapter 2, section 2.4.4) uses regions in the audio that have low or high energy levels to train new speech and silence models on the data that is being processed. The major advantage of this approach is that no earlier trained models are needed and therefore the method is robust for domain changes. The downside of the method is that it is only able to distinguish between speech and silence and not between speech and audible non-speech. In chapter 4 this approach will be extended for audio that contains fragments with high energy levels that are not speech.

Instead of using energy as an initial confidence measure, the proposed SAD subsystem uses the output of an HMM-based broadcast news SAD component. This component is only trained on silence and speech, but because energy is not used as a feature and most audible non-speech will fit the more general silence model better than the speech model, most non-speech will be classified as silence. After the initial segmentation the data classified as silence is used to train a new silence model and a sound model. The silence model is trained on data with low energy levels and the sound model on data with high energy levels and both models are trained solely on data of the recording that is being processed. After a number of training iterations, the speech model is also re-trained using solely the recording. The result is an HMM-based SAD subsystem with three models (speech, non-speech and silence) that are trained solely on the data under evaluation, solving the problem of mismatching training and evaluation conditions.

3.3.2 Segmentation and clustering

Similar to the approach taken for the SHoUT SAD subsystem, the *speaker diarization* system described in [AWP07], is developed to have no tunable parameters or models that are created using a training set (see chapter 2, section 2.4.4). Although this system has proven itself in the rich transcription meeting benchmarks where audio conditions are constant for each recording, the same clustering approach can be used to distinguish between speech from different audio conditions (for example distinguishing between speech from one person speaking either in a studio or outside in the street). The aim not to use tunable parameters or models and the high performance of the system make this a very good approach for clustering data with unknown audio conditions. Unfortunately, this algorithm rapidly becomes slower for long recordings. The majority of processing time is spent on pairwise comparing models. At each merging iteration, all possible model combinations need to be recomputed. The longer a recording is, the more initial clusters are needed. With this, the number of model combinations that need to be considered for merging increases more than linearly.

The diarization step is incorporated into the system in order to make it possible for the ASR subsystem to apply normalization and adaptation techniques on clusters of speech from individual speakers. It is also possible, and more straightforward to apply simple clustering methods in order to obtain clusters for ASR normalization and adaptation. Although these clusters would not contain speech of exactly one single speaker, it has been shown that also these rough clustering techniques help at improving ASR performance. For this research, the diarization approach is preferred above the simple clustering approach because the speaker information obtained by diarization can be a valuable source of information for future research on spoken document retrieval. The information can be used for example for speaker tracking, but also for refining SDR search possibilities.

In chapter 5 the speaker diarization subsystem will be discussed. In this chapter, two approaches to speed up the algorithm are proposed. The first method, referred to as *Cut and Mix*, replaces the pairwise BIC comparisons with a computationally less demanding merging criterion. The second method merges multiple models at each iteration step. More than two models can be merged at once as long as the BIC score for each combination of models is positive.

Also in chapter 5 a thorough analysis of the diarization subsystem is provided. This analysis revealed the performance of each component of the subsystem and this knowledge was used for the 2007 ICSI speaker diarization submission at RT07s [WH08]. This submission together with post-evaluation analysis are also described in depth in chapter 5.

3.3.3 Automatic speech recognition

As shown in figure 3.1, the ASR subsystem consists of four steps. First, feature extraction is performed in a manner so that the features are as much normalized for speaker and audio variations as possible. The results of the first decoding pass are used to adapt the acoustic model for each cluster. These cluster dependent models are then used in the final decoding iteration.

Feature extraction

For feature extraction, two existing techniques were chosen that aim on normalizing the features as much as possible for variation in the audio due to speaker and audio characteristics. A first simple but effective normalization technique that is applied is Cepstrum Mean Normalization (CMN). Vocal Tract Length Normalization (VTLN) is used to normalize for the variation in vocal tract length of the various speakers in both the training set as the evaluation set.

Decoding

The ASR decoder applies a time synchronous Viterbi search in order to retrieve its hypothesis. The Viterbi search is implemented using the token passing paradigm [YRT89]. HMMs with three states and GMMs for its probability density functions are used to calculate acoustical likelihoods of context dependent phones. Up to 4-gram back-off language models (LMs) are used to calculate the priors. The HMMs are organized in a single Pronunciation Prefix Tree (PPT) and instead of copying PPTs for each possible linguistic state (the LM N-gram history), each token contains a pointer to its LM history (see chapter 2, section 2.1). Two issues that were encountered while developing the decoder are discussed in-depth in chapter 6: modularity of the source code and management of the search space.

Because the decoder will be used for research purposes, it is especially important that it is developed to be modular. The code for each type of model should be implemented in its own module. It needs to be possible to easily replace the implementation of each model type by another solution without affecting the source code for the other model types. For example, it should be possible to replace the GMMs that are used as probability density functions by other probability functions without having to re-write the HMM module. In order to assure this modular design, special care is given to how the models interact. For this, an approach similar to [DDCW00] is chosen.

Managing the search space is the second important topic that will be discussed in chapter 6. In order not to run out of memory, the number of tokens in the decoder needs to be kept as small as possible. Various forms of beam pruning and histogram pruning are applied to achieve this. These pruning methods work best when the likelihood of tokens can be compared to each other no matter where they are in the PPT. If the LM probabilities are incorporated at once at the end of the PPT, the likelihood of the tokens passing the end of the tree (the LM calculation) will drop suddenly and it is harder to compare them directly to other tokens (it is not possible to use very tight beams). Therefore Language Model Look-Ahead (LMLA) is implemented [ONEC96]. Using LMLA, the best case LM probability is incorporated at each node in the PPT so that the drop in token likelihood will appear gradually. For decoders such as the SHoUT decoder that do not use PPT copying, implementing LMLA for N-grams of higher order than unigrams, is not straightforward. In chapter 6 the novel method developed for the SHoUT decoder to perform full N-gram LMLA is discussed.

3.3.4 Acoustic model adaptation

The clustering information obtained during segmentation and clustering is used to create speaker dependent acoustic models. The SMAPLR adaptation method [SML00] (see chapter 2, section 2.5.2) is chosen to adapt the means of the acoustic model Gaussians. This method is chosen because it requires hardly any tuning and it automatically determines to what extent the models can be adapted according to how much adaptation data is available. This procedure prevents the models to be over-fitted on the adaptation data when only small amounts of adaptation data are available while it adapts the model parameters as much as possible.

3.4 Summary

In this chapter an overview of the SHoUT system has been given. The system consists of three subsystems: segmentation, diarization and ASR. Each of these subsystems has been designed to contain as few tunable parameters as possible. When tuning is needed, a development set is required that matches the audio that is going to be processed. If the development data are not representable for the target data, the subsystem will be tuned poorly and perform suboptimal. Instead of tuning on a development set, the subsystems will tune itself automatically on the audio that is being processed, so that it becomes possible to process data with unknown audio conditions.

The same mismatch problem exists if statistical models are used. For statistical models, the data used to train the models need to match the conditions of the audio that is going to be processed. Therefore when possible, the use of models that are trained on a training set is restricted as much as possible. This is possible for segmentation and diarization, but not for ASR. For ASR the models are normalized using CMN and VTLN in order to reduce the mismatch between training data and the data that is going to be processed by the decoder.

The subsystems for segmentation, diarization and ASR will be discussed in-depth in the following three chapters.

CHAPTER 4

SPEECH ACTIVITY DETECTION

Speech Activity Detection (SAD) is the task of detecting the fragments in an audio recording that contain speech. Speech activity detection is useful for the ASR subsystem because it is more practical to process small speech segments instead of an entire recording. It is easier to keep the needed computer resources such as processor time and memory usage within reasonable bounds when the length of each segment is limited. A more important advantage of applying SAD is that all non-speech is removed from the recording so that the ASR subsystem doesn't need to process these segments. Although audible non-speech (such as sound effects, etc) do not contain any speech, if they are passed to a decoder it will always output a hypothesis, leading to insertions. SAD is also very important for speaker diarization. All non-speech presented to the diarization subsystem will contaminate the speaker models and this will decrease the quality of the diarization subsystem.[1]

A common approach in speech activity detection is to attempt to classify all types of sound that are present in the recording. If it is known what types of sound can be expected, it is possible to create statistical models for them and the classification is straightforward. SAD is a lot harder when it is unknown beforehand what kind of sound effects can be expected, making it impossible to create high quality non-speech models. In this chapter the research question: 'How can all audible non-speech be filtered out of a recording without having any prior information about the type of non-speech that will be encountered?', is answered and the SHoUT SAD subsystem is presented that is able to handle this task.

The SHoUT SAD subsystem is inspired by the model-based SAD approach described in [AWP07]. During the segmentation process, the models of SHoUT are trained on the audio that is being processed. In order to obtain a bootstrap segmentation that can be used to train these models, in the original algorithm described in [AWP07], a silence-based segmentation strategy is employed (see chapter 2, section 2.3 about segmentation methods). Using this method, no training set is needed to train the models on, and the second research question: 'How can the system per-

[1]The research presented in this chapter is, in part, published in [HWO07]

form speech/non-speech segmentation without the use of statistical models based on training data?', is successfully addressed. When audible non-speech is expected to be present in the audio though, a bootstrap segmentation based on silence will not be sufficient. Therefore, a new solution is needed to solve this research problem. The SHoUT SAD subsystem addresses the problem by applying a model-based segmentation component to create the bootstrap segmentation. After the initial segmentation step, three models are trained on the audio under evaluation: a model trained on silence, a model trained on audible non-speech and a model trained on speech. Each of these models is trained on the data that is being segmented. By applying the three models, the subsystem is able to perform high quality SAD.

In the following section, after discussing the definition of speech and non-speech, the algorithm is described that is used by the SHoUT SAD subsystem. In section 4.3 the features are described and in section 4.4, the two confidence measures are discussed that are needed for deciding which part of the bootstrap segmentation is going to be used to train the new models. In section 4.5 the component that is used to create the bootstrap segmentation is discussed. The bootstrap component is a standard model-based segmentation component for Dutch broadcast news. Finally, in section 4.7 the evaluation of the SHoUT SAD subsystem will be discussed.

4.1 What is considered speech?

When people talk they produce speech, even if what they say is drown out by loud noises or by other speech. For some applications it might be wanted that a speech activity system marks such corrupted speech as actual speech, but when SAD is used as a preprocessing step for ASR, corrupted speech that the ASR subsystem is not able to process correctly anyway, might as well be marked as non-speech. On the other hand, during evaluation, it should be penaltilized if the system is not able to recognize certain parts of the speech. In this thesis, speech is marked as actual speech if the transcriber is able to hear what is being said. The system is expected to process all types of speech as long as a person, the transcriber, is able to understand the content of this speech. Therefore the SAD subsystem needs to be able to classify all speech fragments as actual speech. Even if the fragments contain high levels of noise.

4.2 The algorithm and its steps

The algorithm described in this section aims at training models for the HMM-based segmentation subsystem on the audio it is processing instead of on a separate training set. The subsystem should eventually output a transcription for all speech segments. All other audio events (such as anchor jingles, sound effects or silence) will not be used for further processing, but in order to obtain a speech segmentation, all these fragments need to be marked as non-speech. Therefore, the proposed subsystem trains three models: a silence model, a model for audible non-speech and a model for speech.

In Figure 4.1 the successive algorithm steps are shown. First the audio stream is cut up in chunks of ten minutes. As the number of Gaussians needed in each

GMM is dependent on the amount of data, using chunks simplifies the tuning of the system parameters. In the final algorithm step, the chunks are concatenated. When two neighboring segments from different chunks are assigned to the same class, the segments are merged.

For each chunk, first a bootstrap segmentation is created. This segmentation is used to train models for silence and audible non-speech (first light-gray box in figure 4.1). After training of these models, a model is created for all speech in the recording (second light-gray box). Once all three models are created, it is checked if the audible non-speech model is actually needed. If this is not the case, the non-speech model is discarded and two new models are trained for silence and speech (final light-gray box). In the following subsections, the three steps will be discussed further.

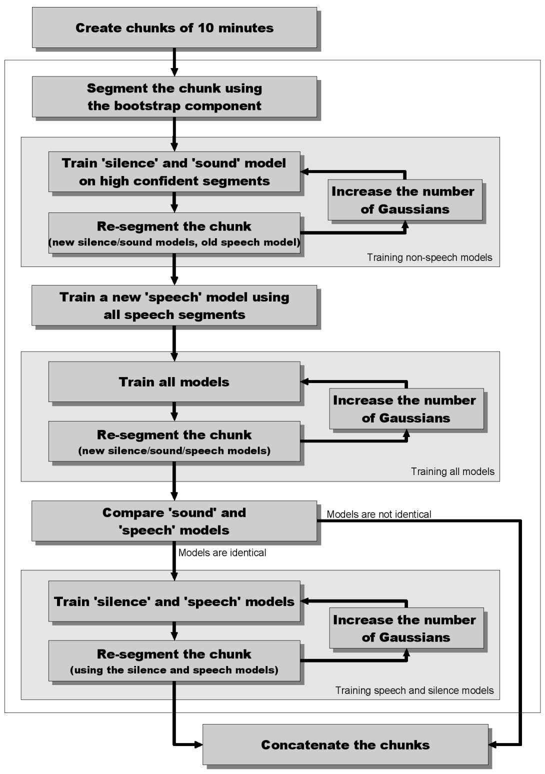

Create chunks of 10 minutes
Segment the chunk using the bootstrap component
Train 'silence' and 'sound' model on high confident segments
Increase the number of Gaussians
Re-segment the chunk (new silence/sound models, old speech model)
Training non-speech models
Train a new 'speech' model using all speech segments
Train all models
Increase the number of Gaussians
Re-segment the chunk (new silence/sound/speech models)
Training all models
Compare 'sound' and 'speech' models
Models are not identical
Models are identical
Train 'silence' and 'speech' models
Increase the number of Gaussians
Re-segment the chunk (using the silence and speech models)
Training speech and silence models
Concatenate the chunks

Figure 4.1: *The algorithm of the speech activity detection subsystem. The audio recording is cut in chunks of* 10 *minute segments and the procedure within the outer box is performed for each chunk.*

4.2.1 Bootstrapping

Each audio chunk is first segmented using a bootstrapping component which segments the data in speech and non-speech fragments. Although the performance of this bootstrapping component does not need to be optimal, it is important that the majority of the data classified as speech actually *is* speech. For the segments classified as non-speech, it is less of a problem when some speech segments are included, as long as most of the silence and sound segments are classified as non-speech.

4.2.2 Training the models for non-speech

Next, a silence and a sound model are created from the part of the data classified as non-speech. Two measures are developed to calculate the confidence that a segment is actually silence or audible non-speech. In section 4.4 these measures will be discussed. All non-speech segments are labeled with the two confidence scores and a small part of the non-speech data that is marked with the highest silence confidence score is used to train an initial silence model. A small amount of data that is labeled with high audible non-speech confidence scores is used to train the initial 'sound' model.

Using these silence and sound models and the primary speech model, a new segmentation is created. This segmentation is used to train silence and sound models that fit the audio very well simply because they are trained on it. All data assigned to the sound and silence models by the new segmentation are merged and any samples that were originally assigned to the speech model in the first iteration are subtracted from the set. This is done to avoid that the sound model starts pulling away all the data from the speech model. This risk is present because although the sound model is already trained on the data that is being processed, the speech model applied is still the old model trained on outside data. Therefore, the sound model may fit *all* of the data better (including speech segments) so that during the Viterbi alignment, speech segments may be assigned to the sound model.

The remaining data is divided over the silence model and the sound model as before. The silence model receives data with high silence confidence scores and the sound model receives data with high audible non-speech confidence scores. This time though, the confidence threshold is not set as high as the first time and consequently more data is available to train each model and therefore more Gaussians can be used to train each GMM. This procedure is repeated a number of times. Although the silence and sound models are initialized with silence and sound respectively, there is no guarantee that sound is never classified as silence. Energy is not used as a feature (see section 4.3) and some sound effects appear to be modeled by the silence GMM very well. Because the goal is to find all speech segments and discard everything else, this is not considered a problem.

4.2.3 Training all models

After the silence and sound models are trained, a new speech model will be trained using all data classified as speech. By now, non-speech will be modelled well by the sound and silence models so that a Viterbi alignment will not assign any non-speech to the speech model. This makes it possible to train the speech model on all data assigned to it and not only on high confidence regions. Once the new speech model is created, all models are iteratively retrained with increasing numbers of Gaussians. At each training iteration the data is re-segmented. Note that in this phase, all data is being used to train the models. During the earlier iterations, the data assigned to the speech class by the bootstrap segmentation component was not used to train the silence and sound models, but because now also the speech model is being retrained, it is less likely that using this data will cause the sound model to pull speech data away from the speech model.

4.2.4 Training speech and silence models

The algorithm works for audio of various domains and with a range of non-speech sounds, but it is not well suited for data that contains speech and silence only. In that case the sound model will be trained solely on the speech that is misclassified at the first iteration (because the initial models may be trained on data not matching the evaluation data, the amount of misclassified speech can be large). During the second training step the sound model will subtract more and more speech data from the speech model and finally instead of having a silence, sound and speech model, the system will contain two competing speech models. Therefore as a final check, the Bayesian Information Criterion (BIC) is used to check if the sound and speech model are the same. As shown in chapter 2, section 2.3.4, the design parameter in the BIC formula that needs tuning on matching data can be omitted when the number of Gaussians in the separate speech and sound models is the same as the number of Gaussians in the combined model. Therefore a new model is created from the data classified as speech *and* from the data classified as sound, with exactly as many Gaussians as the two separate models together. If the ΔBIC score is positive, both models are trained on speech data and the speech and sound models need to be replaced by a single speech model. Again, a number of alignment iterations is conducted to obtain the best silence and speech models.

4.3 Feature extraction

For speech activity detection, Mel Frequency Cepstral Coefficients (MFCC) are frequently used as input feature vectors. Also for the SHoUT SAD subsystem, MFCC is chosen for feature extraction (twelve coefficients). It is common to add energy to the feature vector, but for the SHoUT SAD subsystem, the energy feature is omitted because it will cause audible non-speech to be classified as speech. As will be discussed in section 4.5, the bootstrapping segmentation component is trained on speech and silence but not on audible non-speech. If energy is used, it will play a dominant role in discriminating between the two classes. Because audible non-speech consist of high energy levels (compared to the low levels of silence), audible non-speech will most probably end up in the speech class. For the algorithm described in the previous section it is important that the majority of non-speech, also the audible non-speech, is actually labeled as such and therefore the energy feature is not used.

Although it is not known what kind of audible non-speech can be expected in the evaluation data, it is reasonable to assume that a lot of these sounds will not be generated by a single human voice. In these cases, the *zero-crossing* feature might be a good addition to the MFCC features. The zero-crossing feature is calculated by counting the number of times that the amplitude crosses zero in one frame. It has been shown in [ID71] that for vowels pronounced by humans, the value of this coefficient is only varying within small boundaries while the value can be randomly high or low for other kinds of sounds. Zero-crossing is often used because it does not require complicated and time consuming calculations. In most work, zero-crossing is used in combination with the energy feature.

SHoUT uses the first twelve MFCC coefficients supplemented by the zero-crossing feature. From these thirteen features, the derivatives and the derivatives of these derivatives are calculated and added to the feature vector, creating 39 dimensional feature vectors. Each vector is calculated on a window of 32ms audio and this window is shifted 10ms in order to calculate the next vector.

4.4 Confidence measures

The SAD algorithm described in section 4.2 needs two confidence measures: one for calculating the confidence that a certain fragment is silence and one to determine if a certain fragment is audible non-speech. For the confidence measures used, first all segments that are longer than one second will be split in pieces of one second. The confidence measures will then return a certain amount of one second segments that are most likely to be either silence or audible non-speech.

It is determined if a segment is silence by measuring the energy for each frame and calculating the mean energy of the segment. This calculation is performed for all candidate segments (all segments classified as non-speech by the bootstrap segmentation component) and the resulting values are placed in a histogram. Using the histogram it is possible to return a top amount of segments with the lowest mean energy. As described in section 4.2, a very small amount is chosen for the first iteration and higher amounts are chosen for later iterations.

For determining an amount of segments that is most likely audible non-speech, first the same approach is taken as for silence segments: a top amount of segments is picked with the highest average energy. From these segments a top amount of segments is returned with the highest mean zero-crossing values. In other words, this algorithm returns the segments with the highest mean energy and zero-crossing values. Although audible non-speech segments will have high mean energy values, it is possible that speech segments even have higher average energy values. It is assumed that for these speech segments, the average zero-crossing values will be lower than for the audible non-speech.

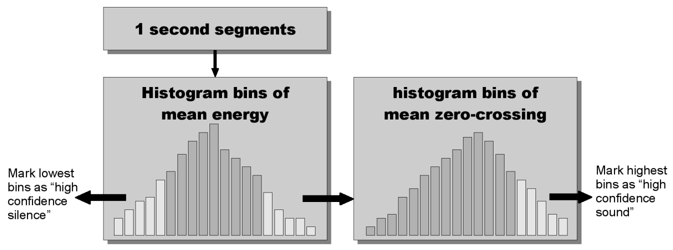

1 second segments
Histogram bins of mean energy
histogram bins of mean zero-crossing
Mark lowest bins as "high confidence silence"
Mark highest bins as "high confidence sound"

Figure 4.2: *A top number of fragments with lowest energy is returned as being silence and a top number of the fragments with highest energy and highest zero-crossing is returned as being sound.*

4.5 The bootstrapping component: Dutch broadcast news SAD

The component that is used to create the initial speech/non-speech segmentation for the SHoUT SAD subsystem is a standard model-based speech activity detection component, developed for finding speech segments in Broadcast News (BN) recordings. As BN shows do not contain a lot of audible non-speech, the component is not trained with any models for music, sound effects or other audible non-speech.

The component consists of an HMM with two strings of parallel states. The first string represents silence and the second string represents speech. The states in each string share one GMM as their probability density function. Using a string of states instead of single states ensures a minimum duration of each segment (see figure 2.9). The minimum duration for silence is set to 30 states (300ms) and the minimum duration for speech is set to 75 states.

The speech and silence GMMs are trained on a small amount of Dutch broadcast news training data from the publicly available Spoken Dutch Corpus (CGN) [Oos00] (see appendix A). Three and a half hours of speech and half an hour of silence from 200 male and 200 female speakers are used. The models are initialized with a single Gaussian. The number of Gaussians is increased iteratively until a mixture of 20 Gaussians is reached for both classes. The data is forced aligned to the reference transcription to ensure the correct placements of speech/silence boundaries. To make sure that only speech is used to train the speech model, all phones neighboring silence are not used.

The BN SAD component uses the feature extraction method described in section 4.3. This means that frame energy is not used as a feature, but zero-crossing is. Because energy is not used, the discrimination between silence and speech will have to be made purely on MFCC features and zero-crossing. It is expected that speech will be well modeled using these features and that any audible non-speech encountered in the evaluation data will fit the general silence model better than the speech model and will therefore be categorized as silence.

If most audible non-speech will be categorized as silence, it is possible to use the

BN component as bootstrapping component for the SHoUT SAD subsystem. In the following section, experiments will show that this is actually the case. It is even possible to use the BN component that is trained on Dutch speech as bootstrapping component for American English speech.

4.6 System parameters

The algorithm for the SHoUT SAD subsystem is developed with the aim to have no parameters that need tuning on a training set. The only 'parameters' left in the system are the silence and speech models trained on broadcast news data. Fortunately, as will be shown in section 4.7 it is possible to use these models for evaluation data that do not match the training data at all and still get good end results.

The system does make use of other parameters such as the number of training iterations performed at each step of the algorithm or the number of Gaussians used to train the models. It is assumed though, that these parameters do not need tuning for specific audio conditions, and that the values of these parameters can be determined using a single development set. Therefore, a development set is created by adding sound effects and some musical fragments to a broadcast news recording. By trial and error the parameters were given their values. The number of Gaussians for each model in each phase of the algorithm are shown in table 4.6. During all following experiments, the parameters were kept fixed at these values.

As can be seen in table 4.6, the number of Gaussians for all models are low at first and increased after each iteration. The final number of Gaussians when the sound model is determined not to be the same as the speech model, will be 7 for the silence model, 16 for the speech model and 18 for the sound model. When the sound is determined to be the same as the speech model and new silence and speech models are being trained, the final number of Gaussians will be 5 for the silence model and 12 for the speech model.

Also the amount of data that is marked as high confidence silence or sound needs to be set. For both the silence as sound models, initially 20 seconds of data are used for each chunk. This 3.33% of the total chunk size is increased by another 20 seconds in the first three iterations of training the two models. In the final two iterations of training the silence and sound models, simply all available data that is marked silence and sound, except for the data that is assigned to speech by the bootstrap segmentation, are used for retraining. In order to obtain the top amount of data with high zero-crossing values, a higher amount of data with high energy levels needs to be selected (see figure 4.2). Therefore, five times as much data with high energy values are selected as are selected for training the sound model. This means that initially, 100 seconds of data with high energy levels are selected and that at each iteration this is increased with another 100 seconds.

Parameter	Value
Initial number of Gaussians for the silence model	2
Initial number of Gaussians for the sound model	2
Number of iterations for training the two models	5
Increase of number of Gaussians after each iteration for the silence model	0
Increase of number of Gaussians after each iteration for the sound model	2
Stop increasing the number of Gaussians after iteration	3
Initial number of Gaussians for the speech model	6
Number of iterations for training the three models together	5
Increase of number of Gaussians after each iteration for the silence model	1
Increase of number of Gaussians after each iteration for the sound model	2
Increase of number of Gaussians after each iteration for the speech model	2
Number of iterations for training the silence and speech models when the sound model is discarded	7
Initial number of Gaussians for the silence model	2
Initial number of Gaussians for the speech model	2
Increase of number of Gaussians after each iteration for the silence model	1
Increase of number of Gaussians after each iteration for the speech model	2
Stop increasing the number of Gaussians for silence after iteration	3
Stop increasing the number of Gaussians for speech after iteration	5

Table 4.1: *The system parameters that were given their values during development. The values of these parameters were kept fixed for all experiments. The parameters are listed here in the order that they were used in the algorithm.*

4.7 Evaluation

The SAD subsystem is evaluated on four different benchmarks. First it is tested on a broadcast news recording. This experiment will provide information about the performance of both the BN SAD component and the entire SAD subsystem. Next, to test system performance on out-of-domain data, the system is evaluated on the RT06s conference meeting evaluation data. Not only are the topics of these meetings different from general broadcast news topics, also the audio conditions and the language do not match the Dutch broadcast news training data (section 4.7.2). A speech/music test set is used to determine if the algorithm is able to classify music as non-speech (section 4.7.3), and finally, twelve fragments from the TRECVID$_{07}$ collection are used for evaluating the system on varying audio conditions (section 4.7.4).

In chapter 2, two metrics for evaluating segmentation systems were described. Both metrics are used in the experiments. The music benchmark will be scored with the metric defined in formula 2.8: the percentage of time frames that are correctly classified. This metric is used so that it is possible to compare the results to earlier work on this collection. The other experiments are scored using the SAD error rate defined by formula 2.9 in section 2.3.5.

4.7.1 Broadcast news evaluation

The broadcast news recording of 27/09/2006 (see appendix A.3) is used to test the SAD system on the BN domain. The SAD error of the bootstrap component is 4.5% on this test set. The error rate of the SHoUT SAD subsystem is also 4.5%. For each chunk, the comparison of the sound and speech models results in discarding the sound model.

4.7.2 Out-of-domain evaluation

In the first out-of-domain evaluation, the SAD component trained on Dutch broadcast news data has been used to determine the initial segmentation. Table 4.2 contains the SAD results on the nine RT06s conference meetings. As a baseline, the error of the bootstrap segmentation coming from the Dutch Broadcast News component, is shown. After the final alignment iteration, the overall error of the baseline on this test set is 26.9% whereas on in-domain Dutch broadcast news it was only 4.5%. This underlines that the conference meeting data is indeed out-of-domain for the bootstrapping models. The overall SAD error of the total system is only 4.4%. This is in line with the state-of-the-art at RT06s. This experiment proves that it is not needed to use a highly performing segmentation component as bootstrap component in order to achieve good results.

file ID	Bootstrap BN SAD	% missed speech	% false alarm	% SAD error
CMU_20050912-0900	42.6	2.8	2.8	5.6
CMU_20050914-0900	41.5	2.3	3.5	5.8
EDI_20050216-1051	13.8	0.6	1.2	1.8
EDI_20050218-0900	16.4	0.8	2.1	2.9
NIST_20051024-0930	20.8	3.8	0.7	4.5
NIST_20051102-1323	17.0	0.8	1.5	2.3
TNO_20041103-1130	32.7	4.5	1.3	5.8
VT_20050623-1400	24.1	1.4	2.3	3.7
VT_20051027-1400	31.7	6.5	1.5	8.0
Overall error	26.9	2.50	1.90	4.40

Table 4.2: *SAD error rates for the RT06s conference meetings*

4.7.3 The IDIAP speech/music evaluation

In the second evaluation, a speech/music test set described in [AMB03] has been used. The data consists of four audio files that contain English broadcast news shows interleaved with various genres of music. The first file contains speech and music fragments of fifteen seconds each. The second file contains fragments of varying lengths but overall with the same amount of speech as music. The third file contains more speech than music while the fourth file contains more music. The performance is measured by (i) the percentage of true speech frames identified as speech, (ii) the percentage of true music frames identified as music and (iii) the overall percentage of speech and music frames identified correctly (see chapter 2, section 2.3.4).

The reference transcripts of this test set only consist of music and speech segments. Any pauses in speech (silence) are not annotated. Therefore, for this evaluation, if a silence segment is neighboring two speech segments, it is merged with these two segments. All other silence and sound segments are labeled as music. This means that silence between speech and music is always labeled as music although it might be the end or beginning of a speech segment.

In Table 4.3 the results of the SAD subsystem on the four files are listed. The SHoUT SAD subsystem does not perform as well as the best system in [AMB03] (on average 95.2%), but considering that it is initialized with Dutch models and that no tuning has been done on a training set similar to this data, the average score of 92.1% can be regarded as satisfactory.

file ID	speech	music	overall
set-1	90.2	95.7	92.9
set-2	88.0	97.0	92.5
set-3	85.1	99.9	92.5
set-4	81.0	99.5	90.3

Table 4.3: *Classification results on the IDIAP speech/music test set. The scores are all percentages of correctly classified frames.*

4.7.4 Dutch TRECVID$_{07}$ ASR evaluation

From the TRECVID$_{07}$ collection, five minute fragments of twelve different documents have been randomly selected (see appendix A). These fragments have been manually annotated and the speech regions are determined by applying forced alignment on the Dutch speech. Table 4.4 lists the results of the system on these twelve fragments. The overall error is 11.4% of the total speech in the audio. Note that only 39 minutes of the in total one hour long test set is actual speech. The bootstrapping BN SAD error is 20.3%. The most part of this error, 15.8%, is due to missed speech and 4.5% is due to false alarms.

4.8 SAD for speaker diarization

When determining the SAD performance, NIST defines a small time region with which the hypothesis segments are allowed to mismatch the reference transcript without being counted as missed speech or false alarms. This so called 'color' of $30ms$ is applied so that small deviations in segment borders that are debatable do not influence the error rate. Because of this, NIST filters small non-speech gaps of less than $30ms$ out of the reference transcript. When these small gaps are not removed from the hypothesis transcript, the gaps will count as errors. Therefore, in order to obtain the optimum

file ID	speech (sec)	BN SAD	% missed speech	% false alarm	% SAD error
15190	274.65	5.9	5.0	1.4	6.4
3273	156.86	38.6	3.9	9.0	12.9
34837	193.59	20.8	11.1	5.3	16.4
3484	196.91	37.2	18.1	0.2	18.2
34973	262.99	7.2	1.7	0.1	1.8
35202	168.71	15.8	4.4	3.0	7.4
35447	204.54	21.5	1.6	7.8	9.4
35757	215.79	16.5	6.8	1.7	8.5
36058	179.62	34.7	15.3	4.5	19.8
36366	73.32	37.4	6.0	15.9	21.9
36626	223.59	17.5	11.5	1.4	12.9
36641	176.06	20.6	15.7	0.1	15.7
Overall	2326.62	20.3	8.3	3.2	11.4

Table 4.4: *SAD error rates for the twelve fragments of the TRECVID$_{07}$ ASR evaluation set. Each fragment is five minutes long. The third column contains the error of the first stage BN alignment.*

score, for SAD evaluation, small gaps are deleted from the hypothesis files. This means that very short pieces of silence are included in the speech segments. For ASR this is no problem because these short silences will be handled by the silence model of the ASR subsystem. In fact, too short speech segments will hurt recognition because there might not be enough context information to perform proper language modeling. During the development of the speaker diarization subsystem though, experiments showed that, although the effect is sometimes little, these small silences do hurt the diarization subsystem (see chapter 5, section 5.4). This subsystem assumes that all input audio is speech from one of the speakers and no silence model is available to remove any short pauses in speech. Therefore any silence in the audio will act as noise during training of one of the speaker models. In order to reduce this source of noise, the SAD segmentation that is passed to the speaker diarization subsystem *should* contain small non-speech segments. Therefore, the segmentation for speaker diarization is not filtered as is done to obtain the best SAD error rate.

4.9 Conclusions and future work

Filtering non-speech out of an acoustically heterogeneous video collection such as the TRECVID$_{07}$ collection is one of the many challenges when automatic annotating the collection. The variety of such collections make it hard to train task specific audible non-speech models. Instead a SAD subsystem was proposed that automatically trains a model for audible non-speech, a so called *sound* model, for each recording in the collection. The system is tested on three benchmarks with promising results.

On the TRECVID$_{07}$ ASR evaluation set, 8.3% of the speech is classified as non-speech. This means that the ASR subsystem will never be able to correctly recognize the words in these regions. On the other hand, using this SAD subsystem, only 3.2% non-speech will be processed by the ASR subsystem. If speech activity detection is not used at all, the percentage of non-speech in the data would be 54% (21 minutes of the total test set is non-speech). Manual inspection of the missed speech showed that most missed speech is speech mixed with various sources of non-speech. It is hard to perform correct ASR on this kind of speech and therefore the loss of being able to process the missing 8.3% of the speech is considered less important than the gain of not needing to process the 54% of non-speech, that would have led to an increase of insertion errors.

The system does contain some system parameters (discussed in section 4.6), but the SHoUT SAD subsystem does not contain any parameters that need tuning on a training set. This makes the algorithm robust for varying audio conditions. It was shown that it is not needed to use a high performing bootstrap segmentation component in order to obtain good final results, and therefore it is not a problem that the speech/silence models used in the bootstrap component are sometimes trained on data mismatching the evaluation data. Having noted this, it would be interesting to investigate other methods to obtain the initial segmentation that do not require any models. One method that might be able to replace the model-based approach is to initially segment on voiced speech fragments. Determining voiced speech regions can be done without the use of models and the majority of the audible non-speech of the resulting segmentation will actually be labeled as non-speech, making it possible to use the proposed algorithm.

A problem related to SAD that is not yet addressed by the proposed system is detecting and discarding foreign speech fragments. Similar to non-speech segments, feeding foreign speech into the ASR subsystem will influence its performance negatively. Unfortunately, as was shown by the experiments on the RT06s conference meeting data, the SHoUT SAD subsystem will classify speech from foreign languages as speech. A solution to this problem is to apply a language detection subsystem directly after the SAD subsystem. Speech from a language for which an ASR system is available can be passed to that system, while speech of other languages can be discarded.

CHAPTER 5

SPEAKER DIARIZATION

Speaker diarization is the task of detecting: 'Who spoke when?'. Speaker diarization systems segment *and* cluster speech on the basis of speaker features. Being able to group all speech from one particular speaker is a useful pre-processing step for various speech processing tasks. For example, an application that summarizes meetings may need to keep track of who said what to whom, and a dialogue act tagger needs utterance boundary information and can exploit speaker change information to model interruptions. Speaker diarization is also useful as an initial step for tracking people across recordings, making it possible, for example, to search for quotes of a specific person in multimedia collections.[1]

The SHoUT ASR subsystem uses the speaker diarization information to optimize its decoding performance. The speaker information is used for normalizing the features for decoding and for adaptation of the acoustic models. For example, vocal tract length normalization and unsupervised acoustic model adaptation (see chapter 2) can be performed for all speech of each speaker. Applying these techniques to speaker clusters is more effective than applying them to the individual speech segments. This will be shown in chapter 6.

As discussed in chapter 3, the aim of each subsystem is to perform its task with as few parameters and statistical models as possible, because parameters need tuning on a development set and models require training on a training set. The less prior knowledge such as training and development data is used, the more robust the system will be against unknown audio conditions. If the system does not contain any tunable parameters and models, no training set is required and this makes it possible to directly use the system for new unseen data without first tuning its parameters and train its models on new data. This approach was taken for the speaker diarization system that has been described in chapter 2. This diarization system is model-based, but its models are created on the audio that is being processed and not on a pre-defined training set so that no training data is required [ABLM02]. In this chapter, after summarizing the algorithm proposed by [ABLM02], the first version of the SHoUT

[1]The research presented in this chapter is, in part, published in [vLH07, WH08, HW07]

diarization subsystem that is based on this algorithm will be described (section 5.2). This subsystem, and the three variants of this system discussed later in this chapter, address the research question: 'How can a speaker clustering system be designed that does not require any statistical models built on training data?'.

The first variant of the speaker diarization subsystem was used during the RT06s diarization benchmark and therefore it will be further referred to as SHoUT$_{D06}$. The SHoUT$_{D06}$ diarization subsystem performed well at RT06s, but not as well as the system described in [AWP07]. In order to answer the question: 'What is the performance of each individual step in the diarization algorithm?', and to improve SHoUT$_{D06}$, a thorough analysis of the system has been performed. This analysis is described in section 5.3.

Using the findings of the analysis, a second diarization subsystem, referred to as SHoUT$_{D07}$, was created and the findings were also incorporated in the ICSI diarization system that participated RT07s. The results of this benchmark and the post-evaluation analysis that were conducted by the author in cooperation with ICSI, will be discussed in section 5.4.

At each merging iteration of both the SHoUT$_{D06}$ and the SHoUT$_{D07}$ diarization subsystems, all speaker models are pairwise compared. Especially for long recordings where a high number of initial speaker models is generated, this procedure is computationally very expensive. Therefore, in section 5.5 the research question: 'How can a speaker clustering system be designed that is able to process long recordings with reasonably computational effort?', will be answered. Two solutions are presented: SHoUT$_{DCM}$ and SHoUT$_{D07*}$.

5.1 Agglomerative model-based speaker diarization

In chapter 2 the speaker diarization algorithm that is used as a baseline for developing the system for this research was discussed in depth. This system was originally developed by [ABLM02] and adopted by the International Computer Science Institute (ICSI). It is model-based, but the models are created on the audio that is being processed and not on a pre-defined training set. The system proved itself in various benchmark evaluations for broadcast news ([AW03, WFPA04]) and rich transcription of meetings ([AWPA06, AWP07, WH08]).

As explained in chapter 2 (section 2.4), an agglomerative algorithm is used. This means that speech data is first divided in a large number of clusters. These clusters are merged pairwise until the correct number of clusters is reached.

Figure 5.1 presents the five steps of the algorithm. In the first step, Speech Activity Detection, all non-speech audio is removed from the data and only speech is used in the remainder of the system. Second, during initialization the speech data is randomly divided in a number of clusters and by iteratively training models for these clusters and re-aligning the speech data, speaker models are created. In the third step, BIC is used to determine which two models are most similar and in the fourth step, if the two models are *not* regarded to be from the same speaker, the optimum number of clusters is reached and the process finishes. Finally if the two models *are* indeed regarded to

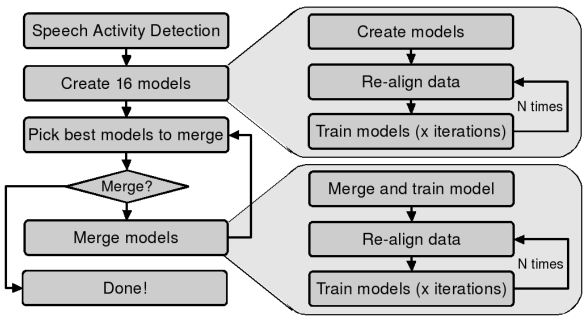


Figure 5.1: *A schematic representation of the speaker diarization algorithm. The steps for creating the initial* 16 *models and merging the models each consist of a number of training and re-alignment iterations.*

be from the same speaker, the two models are merged into one single model and the system resumes at the third step. In the following sections the performance of each of these steps will be investigated.

The *Bayesian Information Criterion* (BIC) is used for both finding the two most similar models and determining when to stop the system (see chapter 2). Using BIC, the similarity of two models is calculated by comparing scores of the two models with the score of a single model that is created using the data of the two separate models. As shown in chapter 2, section 2.3.4, no extra tuning parameters are needed when the number of Gaussians in this single model is equal to the sum of Gaussians in the two separate models. This conforms to the approach of not using any tunable parameters in the system.

5.2 The RT06s submission, SHoUT$_{D06}$

The first speaker diarization system developed for this research is a straightforward implementation of the agglomerative model-based algorithm. It was first tested at the NIST Rich Transcription 2006 Spring (RT06s) evaluation benchmark[2] and therefore this version of the diarization subsystem will be referred to as SHoUT$_{D06}$.

SHoUT$_{D06}$ was implemented using modules available from the SHoUT ASR subsystem. At the time of the evaluation it was not yet possible to create an HMM topology with strings of states sharing a single GMM as drawn in figure 2.9. Therefore in this system, each speaker model consists of a single HMM state. Instead of enforcing a minimum duration of each segment by creating the strings of states, the duration of each segment is influenced by setting the transition probabilities to fixed values. The transition probability from one speaker to another is set to the small value of $\frac{1}{200}$, representing an average segment length of 2 seconds.

[2]RT06s was participated in cooperation with TNO for the AMI project

The system contains five other parameters which are tuned to the RT05s conference meeting evaluation data that is part of the RT06s development data. As shown in chapter 2, section 2.4.4, none of these five system parameters are sensitive for changes in audio conditions [AW03]. In the remainder of this section, a short description of SHoUT$_{D06}$ will be given and the evaluation results will be discussed.

5.2.1 System description

For the first version of the SHoUT speaker diarization subsystem, SHoUT$_{D06}$, the feature extraction component of the Sonic LVCSR toolkit [PH03] is used. This component calculates 12 Perceptual Minimum Variance Distortionless Response (PMVDR) cepstral coefficients. This feature type was developed to be more noise robust than MFCC features (see [PH03]). For speech recognition, energy is added to the twelve coefficients and the first and second derivatives of these features are concatenated to the feature vectors. For this diarization system though, only the PMVDR coefficients are used.

The next step in the algorithm is the creation of the initial clusters. Following the baseline system described in [AWPA06], a fixed number of initial clusters is set. For conference meetings ten clusters are used and for lecture meetings the initial number of clusters is five. Note that the number of initial clusters is fixed for all audio files, no matter how long the files are. The number of Gaussians for each initial model though, is dependent on the total amount of data that is used to train the model. This means that the initial models will contain more Gaussians for longer audio files than for short files. Experiments on the development data showed that this approach outperforms the variant with a fixed number of Gaussians. The optimum number of training samples per Gaussian is 800. For the RT06s conference meetings, this means that each initial model is trained with approximately 10–14 Gaussians. Making the number of Gaussians dependent on the duration of the meeting (the number of training samples) will ensure that the models are not under- or over-trained when the duration of the audio varies. The average number of Gaussians is remarkable higher than the five Gaussians per model reported in [AWPA06]. It suggests that more Gaussians are needed because this system only uses 12 feature coefficients instead of the 19 coefficients in [AWPA06].

The speaker models are all trained in an iterative process. Each model is first trained with a single Gaussian and then the model will split its Gaussian with the highest weight until the desired number of Gaussians is reached. At each iteration, before splitting a Gaussian, all Gaussian means and the covariance matrices will be adjusted in a number of training runs until the overall model score does not improve more than 1.5% relative to the previous training run. After the training of all models, the data will be re-aligned using Viterbi and a new training run (with the existing models) will be started (see figure 5.1). The merged speaker models are created in the same way as the single speaker models. In order to speed up the initialization, the model with the most Gaussians is used as initial model. Then, the model is trained on the data of both speakers and the number of Gaussians is increased iteratively until the correct number is reached.

Finding candidate models to merge and deciding when to stop merging is done with BIC. Figure 5.2 illustrates the merging process. First two models are picked that will be replaced by a single merged model. The remaining models are left unchanged as the merged model is being trained. If there are no two models with positive BIC score that can be merged, the system stops and topology (a) will be the final topology. Otherwise, after training the merged model, all models are trained and the data is re-aligned a number of times. This is also shown in the bottom gray box of figure 5.1.

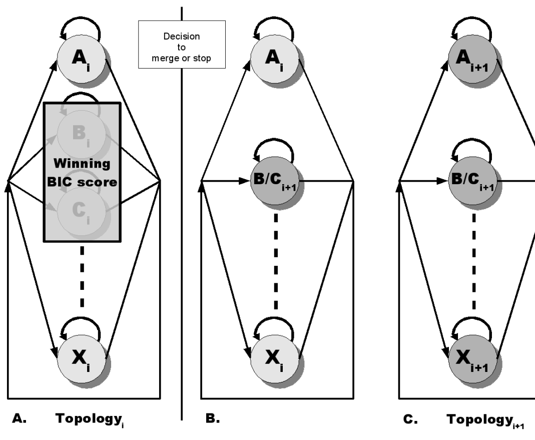


Figure 5.2: *The SHoUT$_{D06}$ system uses BIC to compare models pairwise (a). The two models that are considered most identical (biggest positive BIC score) are replaced by a single model trained on data from both separate models (b). After this replacement, the data is re-aligned and all models are retrained (c).*

5.2.2 RT06s evaluation

The benchmarks for Rich Transcription of meetings contain two speaker diarization tasks (see appendix A). For the Multiple Distant Microphone (MDM) task, multiple microphones are available that are all allowed to be used while for the Single Distant Microphone (SDM) task, only one microphone picked by NIST, is allowed to be used. For the MDM SHoUT$_{D06}$ submission, only the SDM recording is used for feature extraction and the extra information that can be obtained by using multiple microphones, for example beam forming, is not used. The speaker diarization error rates of the diarization system on the conference meeting audio are listed in table 5.1.

Table 5.1 contains the results with and without overlapping speech regions taken into account for calculating the Diarization Error Rate (DER, 2.4.2). It is surprising to see that the DER increases considerably when overlapping speech regions are considered during scoring. Part of this performance degradation is due to the fact that in 2005 and 2006 the speaker segment borders were annotated manually. Especially for overlapping speech regions this may introduce some noise as it is hard to determine

Test set	DER (%) without overlap	DER (%) with overlap
RT05s conference room	21.6	30.2
RT06s conference room	22.7	37.2
RT06 lecture room	30.8	32.4
Processing speed ($\times$RT)	4.63	

Table 5.1: *The speaker diarization results of SHoUT$_{D06}$ measured with and without overlapping speech regions.*

when exactly someone starts or stops speaking. But most part of the degradation can be attributed to the fact that the system is simply not able to model overlapping speech. Because of the Viterbi alignment, all speech is per definition assigned to one single speaker. In the next section it will be analyzed how much of the total DER is actually due to missed overlapping speech.

At the RT06s benchmark the SHoUT$_{D06}$ system performed state-of-the-art and this result shows that the system parameters are tuned sufficiently and that the software is performing adequately. The analysis described in the next section will explain why SHoUT$_{D06}$ was beaten by the ICSI diarization system.

5.2.3 Post evaluation changes

The audio files in the test collection of the RT06s benchmark are all more or less of the same length. Making the number of Gaussians variable helps because the files are not precisely of the same length and the amount of speech in the audio varies for each recording. But for more extreme variations in audio length, keeping the number of initial models fixed will result in models trained on too little data for short recordings and models trained on too much data for long recordings. Models trained on too little data tend to get over-trained and this might prevent models from the same speaker to be selected for merging. Models that are trained on high quantities of data might be so general that all models become similar and are all merged together. In order to prevent these two kinds of mistakes, after the benchmark the system was changed. Instead of making the number of Gaussians variable, the number of initial models was varied and the number of Gaussians was fixed for each initial model. The analysis described in the following section is performed on this new version of SHoUT$_{D06}$.

5.3 SHoUT$_{D06}$ system analysis

In order to get insight in the behavior of the diarization system and the performance of each of the five components, a thorough analysis has been conducted. First, the SHoUT$_{D06}$ system has been compared to the ICSI diarization system. This comparison exposed a number of differences between the ICSI and the SHoUT$_{D06}$ systems. After changing the SHoUT$_{D06}$ system, the performance of the two systems on the test set were in the same range. The new version of the diarization system was used to perform a set of oracle-based experiments [Bin99].

The analysis described in this section focuses on the performance of the diarization system. Although speech activity detection is part of the diarization subsystem, for both the comparison with the ICSI system and for the oracle experiments it is decided to use the SAD component developed by ICSI only. Using the same SAD component for both systems ensures that any variation in DER can be attributed to differences in the speaker diarization part.

5.3.1 RT06s post-evaluation test set

The RT05s test set and the RT06s evaluation set turned out to be relatively small for a proper evaluation of speaker diarization systems (10 and 9 meeting recordings respectively). A small change in the system can be responsible for a big change of the DER of a single recording. If only a few recordings are used as test set, it is possible that the system becomes over-trained for the test set and that the system improvements turn out to be less helpful during the evaluation of other data. Over-training can be avoided by using a test set that is as large as possible.

On the other hand, the larger the test set, the longer it takes to evaluate a system with it. Especially during development this can become a practical problem because the longer an evaluation run takes, the fewer experiments can be conducted. It is possible to evaluate multiple recordings in parallel on different machines in order to reduce the waiting times, but in practice this speed-up is limited as the number of available machines is not infinite.

For the SHoUT$_{D06}$ system analysis, a test set is created that is slightly longer than the RT06s evaluation, but that is short enough to conduct experiments with reasonable high throughput. Table 5.2 contains a list of the twelve conference room recordings that make up this test set. This test set will be referred to as the RT06s post-evaluation test set.

Meeting ID	Meeting ID
AMI_20041210-1052	AMI_20050204-1206
CMU_20050228-1615	CMU_20050301-1415
ICSI_20000807-1000	ICSI_20010208-1430
LDC_20011116-1400	LDC_20011116-1500
NIST_20030623-1409	NIST_20030925-1517
VT_20050304-1300	VT_20050318-1430

Table 5.2: *The RT06s post-evaluation test set: the* 12 *conference meeting recordings used for system analysis.*

All experiments for this analysis have been performed on the Single Distant Microphone (SDM) condition, meaning that only one microphone signal is available for each recording. The Multiple Distant Microphone (MDM) condition was not used, because the speaker diarization system will be applied in a lot of situations where multiple microphone signals are not available. The RT07s submission, discussed in section 5.4, *does* take advantage of the multiple signals in the MDM condition.

5.3.2 Comparison to the ICSI RT06s system

A number of differences were found during the comparison of SHoUT$_{D06}$ with the ICSI RT06s speaker diarization system. The ICSI system uses feature vectors with a dimension of 19 while the dimension of the PMVDR vectors of SHoUT$_{D06}$ is only 12. The number of initial models and the number of Gaussians per model is fixed for the ICSI system. For conference meetings it uses 16 initial models and 5 Gaussians per model. The SHoUT$_{D06}$ system only uses 10 initial models and the number of initial Gaussians is variable. As explained in section 5.2.1, the system will be more robust against variable recording lengths, when not the number of Gaussians but instead the number of models is variable. But, because the recordings in the test set and the evaluation sets are all of the same length, for the analysis, the number of initial models of the new SHoUT$_{D06}$ system is kept fixed at 16 as it is done in the ICSI system. After switching the SHoUT$_{D06}$ system to MFCC vectors of dimension 19 and to 16 initial models, experiments show that the optimum number of initial Gaussians is five.

As described in [AWP07] the HMMs of the ICSI system that represent the single speakers, each consist of a string of states. The use of these strings enforces a minimum duration of each speech segment as there are no transitions that skip states (see figure 2.9). Note though, that during the Viterbi alignment the non-speech fragments are skipped and that these non-speech fragments are later added to the transcription. During this process single speech segments for which the minimum duration is enforced can be cut in multiple shorter segments. The SHoUT$_{D06}$ system only influences the speech segment durations by the value of its transition probability.

The final difference between the two systems is the method used for selecting speech data for each initial GMM. The SHoUT$_{D06}$ system simply cuts the speech data linearly in 16 pieces and uses a piece for each initial model. The ICSI system cuts the data in twice as many pieces. The first half of the pieces is divided linearly over the model and after this, the second half of the pieces is also assigned to the models linearly (see figure 5.3). This way, each model is provided speech originating from two different regions of the recording. The same could be done with three or more batches of data. Figure 5.3 shows how the data can be divided when as many pieces as initial models are used (N=1), when twice as many pieces are used (N=2) or when three times as many pieces (N=3) are cut out of the speech data.

After these findings, the SHoUT$_{D06}$ system was changed so that it is possible to use a string of states for each speaker HMM and to initialize the models using data from multiple speech regions. The diarization error rate of SHoUT$_{D06}$ before changing either of these two aspects (but after switching to the MFCC features) was 14.70% while the error rate of the ICSI system is 11.74% DER. Tuning experiments show that for the SHoUT$_{D06}$ system, the optimum number of states per HMM is 250 (a minimum duration constraint of 250ms). Enforcing the minimum duration reduces the diarization error rate by 1% absolute to 13.70%. Although no theoretical advantage can be thought of to initialize the models by cutting up the data in twice the number of pieces as there are initial models (N=2 in figure 5.3), the SHoUT$_{D06}$ system has been changed to make this possible resulting in a further drop of the DER

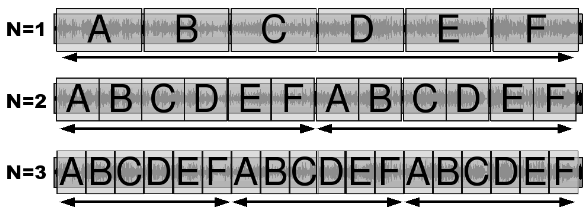


Figure 5.3: *This figure illustrates how a speech recording is divided into pieces for training the initial models of the speaker diarization system. In this example data for* 6 *initial models are needed (blocks A to F). It is possible to select the data continuously from the recording (N=1) and to train each model with one piece. It is also possible to create multiple smaller pieces (N=2,3,etc) and train each model with two or more of those pieces.*

to 11.69%.

It should not matter how the data is divided over the models for initialization because it is unknown beforehand how the speakers are distributed in the data. Whether method N=1 or N=2 is used, the models are provided with unknown randomly chosen initial data. It might be possible that the N=2 method is favorable because of some typical feature of the speaker distribution in meetings, but this possibility has not been investigated. Instead, the following experiments focus on the second parameter that is important during initialization: the number of iterations for training the models. As depicted in figure 5.1 (top-right gray box), during initialization of the models, the data is re-aligned N times and the models are trained with x training iterations. In the ICSI system, $N = 3$ and $x = 5$. In the SHoUT$_{D06}$ system, N is also set to three, but it seems that this system needs more training iterations and therefore, x is set to 20. The number of training iterations was originally dependent on how well the models were improving during each iteration. Instead of a fixed number of times, the models were trained until a training iteration did not improve the model with more than 1.5%. Experiments show that in general not more than 20 iterations are needed for this and therefore for comparison with the ICSI system, the setting $x = 20$ has been chosen.

	Number of training iterations			
	5	10	20	30
DER with N=1	14.30	14.32	13.70	11.76
DER with N=2	14.67	12.22	11.69	11.89
DER with N=3	14.87	14.79	11.89	11.64

Table 5.3: *Diarization error rates using the three methods of data initialization with various number of training iterations. For all initialization methods, the DER decreases with the increase of the number of training iterations.*

In table 5.3 the diarization error rate is listed for the three initialization situations $N = 1$, $N = 2$ or $N = 3$ (see figure 5.3) and with a varying number of training iterations. The trend in this table is that the DER decreases when the number of training iterations increases. Twenty training iterations is sufficient for $N = 2$ and $N = 3$, but on this test set for $N = 1$ more iterations are needed. Even though the test set is chosen as big as possible, it is still noted that small random changes to the system can influence the DER significantly. Experiments with more data are needed to prove whether the outlier of 13.70% when 20 iterations are used with $N = 1$, is due to such a random change or that the initialization method with $N = 1$ actually *does* need more training iterations and therefore is less efficient than $N = 2$ or $N = 3$. But because no theoretical reason is available why $N = 2$ should be better than $N = 1$, the only conclusion drawn from these experiments is that in general, the DER decreases when the number of training iterations is increased. The risk of over-training the models when more iterations are used is minimal because the training data is the same data as the evaluation data. The only disadvantage of using more training iterations is a slower system. Fortunately the system speed is still manageable using 30 training iterations and therefore the number of training iterations is increased from 20 to 30. The initialization method (N=1) is not changed.

Meeting ID	% SAD error	% Spkr error	DER
AMI_20041210-1052	1.60	7.10	8.71
AMI_20050204-1206	4.80	4.50	9.27
CMU_20050228-1615	10.50	4.80	15.28
CMU_20050301-1415	5.40	1.90	7.27
ICSI_20000807-1000	5.00	4.00	9.04
ICSI_20010208-1430	4.80	12.80	17.61
LDC_20011116-1400	5.10	4.40	9.45
LDC_20011116-1500	7.00	7.90	14.93
NIST_20030623-1409	1.70	2.70	4.31
NIST_20030925-1517	13.40	12.40	25.79
VT_20050304-1300	1.60	3.10	4.72
VT_20050318-1430	7.50	16.40	23.87
Overall	5.30	6.40	11.76

Table 5.4: *The results of the diarization system on the test set of twelve conference meetings. The Diarization Error Rate (DER) is the sum of the SAD error and the error due to classifying speech as the wrong speaker (spkr error).*

The comparison of the SHoUT$_{D06}$ system with the ICSI diarization system revealed a number of shortcomings of the SHoUT$_{D06}$ system. Enforcing a minimum segment duration by using multiple states for each speaker HMM and increasing the number of training iterations improved the system. The DER of 11.76% is in the same range as the error rate of the ICSI system (11.74%). The performance of the SHoUT$_{D06}$ system is listed in table 5.4. In the remaining of this section, the SHoUT$_{D06}$ system will be further analyzed and for each step in the algorithm it will be determined what its contribution to the DER is.

5.3.3 Oracle experiments

The remaining analysis will be based on oracle experiments. Oracle experiments are 'cheating' experiments where the system or part of the system can make use of whatever knowledge is available. Even the optimal system output, the reference transcripts, may be used. In a sense, the system is an oracle that knows everything [Bin99].

In this section, oracle experiments are used to assess the performance of separate system components. For each of these tests, the components that are *not* tested are replaced by an experimental setup that performs optimally by using knowledge of the reference transcription (the oracle knowledge) and the components that *are* tested are left unchanged. At first all components will be replaced by the oracle setup and the DER will be measured. Then, one at a time, the actual components are placed back into the system. The increase in DER after replacing a component is attributed to shortcomings of that particular component. After all components are placed back into the system, a part of the DER of 11.76% can be attributed to each component. This method is depicted in figure 5.4.

The same test set used in the previous experiments will be used for the oracle experiments (table 5.2). The diarization error rate (DER) of 11.76% (see table 5.4) is calculated using the NIST metrics and the total diarization error is the sum of two rates: the Speech Activity Detection (SAD) error and the speaker classification error. The SAD error is the percentage of speech and non-speech that is misclassified. The speaker classification error is the error due to misclassifying speakers [FA07]. The various oracle experimental setups that replace the actual components all make use of the reference transcripts in one way or another. Before discussing the experimental setups and the experiment results, it is first described how the reference transcripts are used.

5.3.4 Reference transcripts

For scoring the output of diarization systems, a reference transcript is manually created labeling the fragments that each speaker is talking [FAG08]. In the oracle experiments, these reference transcripts are used in three different ways. In the first experiments of the series, the transcripts are used as input for the diarization system. One way of doing this is to replace the SAD output with the transcription. In this case the IDs of all speakers in the transcripts are replaced with one ID ('speech') and all overlap regions are replaced by single regions. Another method for using the reference transcripts as input, is to replace the initial clustering with a perfect clustering obtained from the reference transcript.

The reference transcripts are also used to take merging decisions. Instead of performing BIC, the oracle merge component will score a segmentation with the NIST scoring tools[3] and determine for each cluster which speaker it represents most (which speaker is classified by that cluster the longest period of time). The *purity* of each cluster is then calculated as follows: The purity of cluster A is the time that the representing speaker was classified as A divided by the total amount of time that was

[3] `http://www.nist.gov/speech/tools`

classified as *A* (see chapter 2, section 2.4.2). The oracle merge component will then decide to merge the two clusters with the same representing speaker that have the highest purity.

Third, the system is modified so that an intermediate hypothesis segmentation file can be printed after each merging iteration. The reference transcript is then used for scoring these segmentations for monitoring purposes.

5.3.5 Experimental set-up, six oracle experiments

In total, six oracle experiments have been conducted. In the first experiment, all algorithm steps are replaced by an experimental setup that uses oracle information, and at each following experiment one of the components is placed back into the system. This procedure is depicted in figure 5.4. Next, the six experimental setups are described.

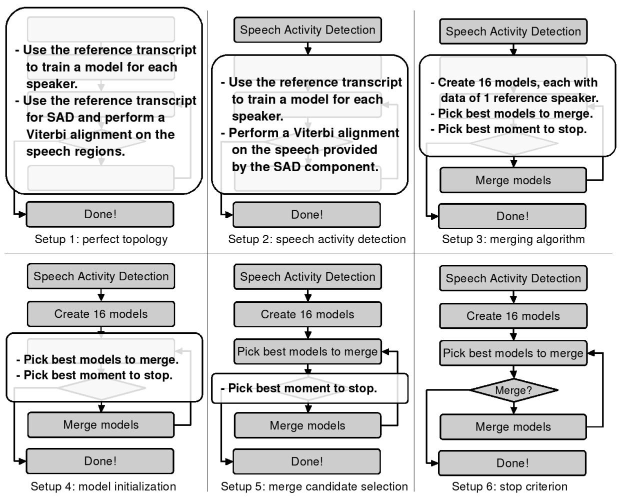

- Use the reference transcript to train a model for each speaker.
- Use the reference transcript for SAD and perform a Viterbi alignment on the speech regions.
Done!
Setup 1: perfect topology
Speech Activity Detection
- Use the reference transcript to train a model for each speaker.
- Perform a Viterbi alignment on the speech provided by the SAD component.
Done!
Setup 2: speech activity detection
Speech Activity Detection
- Create 16 models, each with data of 1 reference speaker.
- Pick best models to merge.
- Pick best moment to stop.
Merge models
Done!
Setup 3: merging algorithm
Speech Activity Detection
Create 16 models
- Pick best models to merge.
- Pick best moment to stop.
Merge models
Done!
Setup 4: model initialization
Speech Activity Detection
Create 16 models
Pick best models to merge
- Pick best moment to stop.
Merge models
Done!
Setup 5: merge candidate selection
Speech Activity Detection
Create 16 models
Pick best models to merge
Merge?
Merge models
Done!
Setup 6: stop criterion

Figure 5.4: *The six experimental setups. The yellow boxes represent the oracle components that replace the actual components. The oracle components perform their task using the reference transcript. In each experiment, one component is placed back into the system.*

Perfect topology

If the algorithm would do a perfect job, the HMM would contain exactly one model per speaker and each model is trained on all the available speech of its speaker. Even if the algorithm would not make a single mistake and this perfect topology was created, the system is not expected to have a perfect diarization score because the system is not able to model overlapped speech and because the models, with their limited number of Gaussians, might not be able to classify all speech perfectly. In order to test the system on these limitations, in the first experiment the reference transcription is used instead of the SAD component. For each speaker in the reference, one model is trained using all speech from that particular speaker. The total number of Gaussians in the system is the same as normal (80) and they are divided over the clusters based on the amount of speech that is available for each speaker. After the models are created, a Viterbi pass is performed to find the final system result.

Speech activity detection

In the second experiment, the actual SAD component is placed back into the system. The models are still trained directly on the speech from the reference transcription, but the final alignment is performed with the actual SAD segmentation. The increase of error rate compared to the previous experiment is the amount of the total system error that can be blamed on the SAD component.

Merging algorithm

Next it is tested what the influence of using the actual merging algorithm is on the final result. For this test, the reference transcript is used to create sixteen initial models. Each model is created with speech of only one speaker, but because now sixteen models are needed, the speech of each speaker is cut up in pieces so that multiple models can be trained for each speaker. The data are divided so that each model is trained on an average amount of data (a person that spoke a lot in a meeting, will have a high number of initial models). The normal model initialization and merging procedure are used, but the decisions about which models to merge and when to stop are performed by the oracle setup (as described in section 5.3.4). The DER scored on this experiment subtracted by the error of the previous experiment will reveal the error that is introduced by the procedure of creating the final models by merging the smaller initial models together.

Model initialization

Instead of creating the initial models with use of the reference transcript, in this experiment the initial models are created normally by dividing the speech data randomly. The merging and stop decisions are still performed by the oracle setup though. Therefore the increase of error can be attributed to the shortcomings of the systems model initialization method.

Merge candidate selection

The oracle merge candidate selection component is now replaced by the actual candidate selection component based on BIC. The increase of DER between the previous experiment and this experiment can be attributed to the shortcomings of the BIC selection component.

Stop criterion

Finally, the remaining oracle setup that is able to decide when to stop merging perfectly, is replaced by the actual component that decides when to stop based on BIC. The difference between the actual system DER and the previous experiment will reveal the error gained because of incorrect stop decisions.

5.3.6 Experiment results

The six oracle experiments are performed on RT06s post-evaluation test set of twelve conference meetings. The results of these experiments are listed in table 5.5. In the first experiment, the entire algorithm to create the HMM topology is bypassed and the models are created directly. At each following experiment, one step of the algorithm is placed back into the system. Assuming that the components are mostly performing independent of each other (see section 5.3.7), at each step, the increase in DER is a good indication of the contribution to the total error of the component placed back.

Oracle experiment	SAD (%)	DER (%)
Perfect Topology	2.60	4.18
Speech Activity Detection	5.30	6.87
Merging Algorithm	5.30	7.36
Model Initialization	5.30	9.18
Merge Candidate Selection	5.30	10.40
Stop Criterion	5.30	11.76

Table 5.5: *The SAD and DER errors of the six oracle experiments. The experiments are named after the description titles in section 5.3.5*

Although in the first experiment, the reference SAD is used, the SAD error is still 2.60%. This error is due to the inability of the system to model overlapped speech. The total DER in this experiment is 4.18%. That means that although the merging algorithm is bypassed, the modeling approach is not perfect and is responsible for 1.58% DER. In the second experiment, where the actual SAD component is used, the SAD error increases to 5.3%. Part of this error is because of overlapped speech (2.6%) but 2.7% is due to errors in the SAD component. The increase in DER in the third to the sixth experiments can be attributed to the use of the merging algorithm (0.49%), initializing the clusters (1.82%), performing BIC to combine models (1.22%) and determining when to stop merging (1.36%). Table 5.6 summarizes the contribution to the DER of each component.

As can be seen in table 5.6 the three factors that contribute most to the total DER are the lack of being able to model overlapped speech, the speech activity detection itself and the initialization of the sixteen clusters.

Test description	DER (%)	Relative
Overlapping speech	2.60	22.11
Speech Activity Detection	2.70	22.96
Modeling/alignment	1.57	13.35
Merging algorithm	0.49	4.17
Non-perfect initial clusters	1.82	15.48
Combining wrong models	1.22	10.37
Stop clustering too early/late	1.36	11.56
System DER (sum of components)	11.76	100.00

Table 5.6: *The contribution of each system step to the overall DER.*

5.3.7 Discussion and conclusions

In this section, the SHoUT$_{D06}$ speaker diarization system that trains its GMMs only on the data under evaluation according to a five step algorithm, was analyzed by performing a series of oracle experiments.

During this analysis it was assumed that the performance of each component is mostly independent of the performance of others. Unfortunately though, changing one component *is* likely to have an impact on other components. Blaming a single component for the increase of DER between two experiments, as was done in section 5.3.6 might sometimes give a slightly distorted picture. A way to investigate the dependencies between components is to test each component with input of varying quality. The results of such experiments would probably provide extra information and possibly nuance the current results.

It should be noted that the results of this analysis are partly dependent on the evaluation data and that a high contribution to the error of a certain component does not necessarily mean that it is possible to improve this component. Also, the results apply to the analyzed diarization system and might be different for other systems. In spite of these caveats though, this type of analysis provides valuable insight of the component behavior and guidance for improving the system.

One of the components that contributed most to the total DER is the speech activity component. This triggered the development of the SHoUT SAD subsystem (chapter 4). Experiments in the following section will show that using this SAD subsystem improves the speaker diarization system considerably.

5.4 The RT07s submission, SHoUT$_{D07}$ and ICSI-RT07s

The analysis described in the previous section revealed a number of shortcomings of the speaker diarization system. These shortcomings are addressed in a new version of the system that will be referred to as SHoUT$_{D07}$. As described earlier, feature extraction has been changed from PMVDR vectors with 12 dimensions to MFCC vectors of 19 dimensions, minimum duration enforcement has been added and the number of training iterations have been tuned. After the oracle analysis, also a new SAD subsystem was developed. This subsystem was described in-depth in chapter 4.

Also the ICSI diarization system was equipped with this new SAD component and for the RT07s speaker diarization submission, referred to as ICSI-RT07s, research forces were joined. New experiments were conducted, both before and after the evaluation, to find the optimal system performance. These experiments, discussed in this section were conducted on the ICSI-RT07s system. This section describes the ICSI-RT07s speaker diarization system shortly and then focuses on these experiments.

5.4.1 The ICSI-RT07s speaker diarization system

Except for differences in implementation choices, the most outstanding difference between the SHoUT$_{D07}$ system and the ICSI speaker diarization system is that the SHoUT$_{D07}$ system does not profit from available multiple microphone signals. The ICSI system profits from multiple microphone signals in two ways. First the MFCC features are created from an enhanced signal obtained by beam-forming the multiple signals. Second, the delays between microphone pairs calculated for the beam-forming procedure, are used as a second feature stream. The model scores of the two feature streams are added using a weight for each stream to obtain the final model scores. These weights are determined using the algorithm introduced in [AWP07]. This approach makes it possible to automatically find appropriate weights for the two streams during the diarization process and eliminates the need to tune the weights on a development set. Further information on the ICSI diarization system can be found in section 2.4.4 or in [AWPA06, AWP07, WH08].

In the remainder of this section, the ICSI-RT07s evaluation results and the results of the development experiments will be discussed. All experiments conducted in the remainder of this section are performed on the multiple distant microphone task.

5.4.2 Test set

Table 5.7 lists the names of the 21 meetings that are used for development testing for the RT07s evaluation. Because of the ‘flakiness’ of the diarization error rate [MW06], as many meeting recordings as possible are used for development. This helps to ‘smooth’ diarization error rates by preventing large variations in the scores of one or two meetings from influencing the overall DER (see 5.3.1). Note that this test set is even bigger than the set in section 5.3.1.

All of the experiments reported here are conducted using data distributed by NIST as part of the Rich Transcription 2004, 2005, 2006 and 2007 meeting recognition evaluations. This data consists of excerpts from multi-party meetings collected at eight different sites. From each meeting, an excerpt (chosen by NIST) of 10 to 15 minutes is used.

Meeting ID	Meeting ID
ICSI_20000807-1000	ICSI_20010208-1430
LDC_20011116-1400	LDC_20011116-1500
NIST_20030623-1409	NIST_20030925-1517
AMI_20041210-1052	AMI_20050204-1206
CMU_20050228-1615	CMU_20050301-1415
VT_20050304-1300	VT_20050318-1430
CMU_20050912-0900	CMU_20050914-0900
EDI_20050216-1051	EDI_20050218-0900
NIST_20051024-0930	NIST_20051102-1323
TNO_20041103-1130	VT_20050623-1400
VT_20051027-1400	

Table 5.7: *The names of the* 21 *meetings used for development.*

5.4.3 Speech activity detection

The speech activity detection system described in chapter 4 outperformed the ICSI-RT06s SAD system on the development set. Although typically in the meeting domain the number of non-speech sounds is negligible, in two of the twenty one meetings of the development set, the system classified part of the audio as non-speech sounds (paper shuffling and doors slamming). In the other meetings, (including all of the meetings in the RT07s evaluation set) the BIC score was always positive and each sound model was discarded.

Table 5.8 contains the results of the ICSI-RT06s SAD system and the SHoUT segmentation subsystem on the test set. The first two rows show the results scored only for speech activity detection. The new system has a slightly lower false alarm rate. The last two rows of table 5.8 show the results of the ICSI-RT07s diarization system using either the SAD segmentation of the RT06 detector or the RT07 detector. On this data, the new detector has a lower false alarm rate. Most of the performance gain though is a result of the reduction in speaker error (diarization). This is partly explainable because the SAD segmentation is not smoothed before diarization (see the next experiment). It is surmised that the remainder of the gain is due to the reduced number of false alarms and it seems likely that this helps to make the data used to train the GMMs 'cleaner', resulting in better models.

5.4.4 Smoothing SAD

Most SAD systems, including the ICSI-RT06s SAD system, are tuned by minimizing the SAD error on a development set. One of the steps that helps in minimizing the SAD error is 'smoothing' the output (NIST provides scripts to do this). During this process, short non-speech segments (shorter than 0.3s) are removed from the segmentation. Smoothing helps to reduce the SAD error because the reference segmentation is smoothed as well, and so these short fragments of non-speech will be

System	% missed speech	% false alarm	% SAD	% Spkr	% DER
RT06 (only SAD)	1.10	2.80	3.90	n.a.	n.a.
RT07 (only SAD)	1.20	2.10	3.30	n.a.	n.a.
RT06 (diarization)	4.40	2.30	6.70	4.10	10.81
RT07 (diarization)	4.50	1.50	6.00	2.50	8.51

Table 5.8: *Performance of the RT06 and RT07 speech/non-speech detectors on the RT07s Eval data. The detector used for RT07 is the SHoUT segmentation subsystem, described in chapter 4. In the first two rows, only the SAD segmentation is scored. The last two rows show the results of the RT07 diarization system using either the RT06 speech/non-speech system or the RT07 speech/non-speech system.*

regarded as missed speech if no smoothing is performed. On the other hand, adding these short non-speech segments to the speech data that is processed by the speaker diarization system will most likely increase the DER. The non-speech will be assigned to one or more clusters and will ‘muddy’ the data pool, forcing the GMMs to be less specific for a particular speaker. Therefore, for RT07s it was decided to use the unsmoothed speech/non-speech segmentation as input to the diarization system and perform smoothing after the diarization process is finished. The improvement over using the smoothed speech/non-speech segmentations on the test set is marginal. On the conference room MDM task, using the smoothed segmentation results in a diarization error of 9.03%, and so the improvement by using the unsmoothed speech/non-speech input is only 0.52% absolute.

5.4.5 Blame assignment

In order to find out what part of the diarization system is contributing most to the total DER, some more cheating experiments, similar to the oracle experiments in section 5.3 have been conducted. Instead of using the automatically generated speech/non-speech segmentation, the reference segmentation is used as input for the diarization system. Table 5.9 contains the error rates of the MDM and SDM submissions and the results of the oracle experiments. All results are scored with and without overlap.

Even if the reference segmentation is used, the percentage of missed speech will not be zero. This is because the diarization system is only able to assign a speech fragment to one single speaker and thus, when scoring with overlap speech, all overlapping speech will be missed. As can be seen in the second row of table 5.9 the error due to missed overlapping speech is 3.7%. The total error due to missed speech and false alarms is 6.0%. Subtracting the error due to overlap leaves the error contribution of the SAD system: 2.3%. The remaining 3.8% of the total DER is caused by the diarization step (speaker error). Note that the percentages change slightly if scored without overlap because ignoring segments with overlap will decrease the total amount of speech, which is part of the DER calculation. The same blame assignment can be

	%Miss	%FA	%Spkr	%DER
MDM -ref +ovlp	**4.5**	**1.5**	**2.5**	**8.51**
MDM +ref +ovlp	3.7	0.0	3.8	7.47
MDM -ref -ovlp	0.9	1.6	2.6	5.11
MDM +ref -ovlp	0.0	0.0	3.9	3.94
SDM -ref +ovlp	**5.0**	**1.8**	**14.9**	**21.74**
SDM +ref +ovlp	3.7	0.0	12.8	16.51
SDM -ref -ovlp	1.4	2.0	14.7	18.03
SDM +ref -ovlp	0.0	0.0	12.7	12.75

Table 5.9: *DER for the MDM and SDM submissions. The rows in bold show the results of the actual submissions. They are scored with overlap and make use of the speech/non-speech segmentation. The systems marked with -ovlp/+ovlp are scored with/without overlap and the systems marked with -ref/+ref make use of the automatic speech/non-speech segmentation or of the reference speech/non-speech segmentation.*

done for the SDM task. The error because of missed overlapping speech for the SDM task is 3.7%, and the error due to the SAD component is 3.1% (3.4% if scored without overlap). The speaker error caused by the diarization system is 14.9%. Note that the error due to missed overlapped speech is higher than found in section 5.3, because this set contains more overlapped speech.

5.4.6 Noise Filtering

In a series of experiments, it has been tested how much the system gains from applying Wiener filtering. Wiener filtering is normally applied to the audio used for the SAD component, on the audio that is used to create MFCC features, and on the audio that is used to calculate the delay features. Table 5.10 shows the results of several experiments where filtering is omitted for one or more of these components. It shows that filtering helps to reduce the DER considerably. Although it seems that filtering the audio for speech/non-speech helps the most, the SAD error on unfiltered audio only increases marginally (from 3.3% to 3.4%).

Where do we apply Wiener filtering?	%DER
Nowhere	15.80
Speech/non-speech	10.54
Speech/non-speech and MFCC	12.99
Speech/non-speech and Delays	13.70
All components	8.51

Table 5.10: *DER for the MDM submission (bottom row) and for the experiments where Wiener filtering is omitted in one or more of the components.*

5.4.7 Delay Features

The algorithm introduced in [AWP07] is used to automatically determine stream weights for the MFCC and delay feature streams. In the RT06s submission, the weights were fixed to 0.9 and 0.1. An experiment is conducted on the RT07s evaluation data where the weights are also fixed (as was done for RT06s) in order to determine if the adaptive weighting was the right choice for the evaluation data. On the MDM conference meeting task the DER is 9.29% using the RT06s fixed weights. Thus, the new algorithm improves the DER by 0.78% absolute.

The gap between the results in the SDM task and MDM task is considerably large. This performance difference could be because it is not possible to use the second (delay) feature stream for SDM. To test this hypothesis, the system is applied on the MDM data using only the MFCC stream. The diarization error of this experiment is 14.02%. A difference of 5.51% DER absolute.

5.4.8 Discussion

In this section, experiments on the ICSI-RT07s speaker diarization system were presented. The SHoUT segmentation subsystem (chapter 4) that is able to filter out audible non-speech without the need for models trained on 'outside' data reduced false alarm errors by 0.7% absolute compared to the RT06s speech/non-speech system. Post-evaluation experiments showed that by reducing the false alarms, the diarization system also performed better (2.3% DER absolute).

Other post-evaluation experiments showed that the use of cross-channel delays as a second feature stream (for the MDM task) improved the system considerably resulting in a gain of 5.51% DER absolute. It was also observed that omitting noise filtering in either one of the feature streams decreases the performance of the system by up to 7.29% absolute. Modest improvements were obtained in system performance by using unsmoothed speech/non-speech segmentations as input to the diarization system (0.52% DER absolute). Another modest improvement was achieved by dynamically tuning the stream weights as proposed by [AWP07] rather than using fixed stream weights (0.78% DER absolute).

The gap in performance of the system between the SDM and MDM tasks is striking. The post-evaluation experiments showed that the errors due to missed overlapping speech and misclassified speech/non-speech are comparable for the two tasks. Thus, the main difference in performance is caused by the diarization system itself (3.8% DER for MDM and 14.9% DER for SDM). It seems likely that the SDM system can be improved considerably by introducing additional feature streams, similar to what was used in the MDM system. Of course these additional streams would not be based on delays since there is only a single microphone in the SDM condition, but other acoustic features could be used (e.g. PLP or RASTA features), or even the output of other speaker diarization systems as additional feature streams.

Finally, because of the 'flakiness' of the diarization error rate, all experiments discussed in this section were performed using a much larger set of recordings (21 in total) than used in past evaluations. Using this larger set of data helps to reduce some of the flakiness, thus leading to better decisions about system design and tuning.

5.5 Speaker diarization for long recordings

A potential disadvantage of the agglomerative approach using BIC, is that at each comparison step, a new model needs to be trained that contains as many Gaussians as the two separate models. Training this merged model for all combinations of speakers takes a considerable amount of processing time. For recordings that are about half an hour long, the needed time for training these models is manageable, but for longer recordings, needed processing time becomes a critical issue. If the audio file is very long, a high number of initial clusters needs to be created and an even higher number of merged models needs to be trained at each iteration step. The number of initial models increases linearly with the length of the audio file and therefore more initial models will not influence the real-time factor. Unfortunately, the needed number of merged models increases with power two compared to the number of initial models, making the overall complexity of the algorithm $O(n^3)$. This means that for long audio files so many merged models need to be created at each iteration step that the processor time needed to create them will become critical.

One way of speeding up the creation of these merged models is to simply write very time efficient source code for the parts that are computationally expensive and used frequently. For example, it is beneficial to replace the standard code for performing logarithmic calculations with fast but accurate look-up tables [VFM07]. Unfortunately, creating efficient code in general will make the system less flexible for changes. As SHoUT$_{D07}$ is developed for research purposes, readability and the possibility of changing parts of the algorithm are considered more important than speed issues. Therefore, the source code is not yet optimized for speed. It is possible that by writing efficient source code, the system will become able to process longer audio files.

Another and more constructive way of solving the speed issue for long audio files is to alter the algorithm. Even when using the most efficient code, the maximum duration of audio files that can be processed is limited because of the $O(n^3)$ complexity of the current algorithm. In this section, two variants of the algorithm will be discussed that are designed to speed up the system. The first approach replaces the local pairwise BIC comparison for a global system-wise comparison and in the second approach multiple models are merged per iteration.

5.5.1 The Cut&Mix system, SHoUT$_{DCM}$

Considering the merging of all combinations of two models, as described in the sections 2.3.4 and 5.1, takes a considerable amount of computational effort. Another disadvantage of SHoUT$_{D06}$ and SHoUT$_{D07}$ is that they are based on the assumption that each pair of two models is trained with data from the same single speaker. Unfortunately when a model contains data from multiple speakers, this data is not spread over multiple models when the model is taken out of the topology. The system introduced here, that will be referred to as SHoUT$_{DCM}$, does not compare or merge the models pairwise and is computational less demanding than the original diarization subsystem.

System description

The basic idea of the SHoUT$_{DCM}$ strategy is that all models are considered for removal, and that the model which improves the Viterbi likelihood most after removal is subsequently 'cut out', and its Gaussians are redistributed over the remaining models. Formally, at each iteration i the number of models in the HMM will be reduced by one, by sequentially removing model j. The overall Viterbi score L will be used to compare the original HMM topology $\mathcal{T}_i$ with the new topology $\mathcal{T}_{i+1}^j$. If the score $\max_j L(\mathcal{T}_{i+1}^j)$ is higher than the original score $L(\mathcal{T}_i)$, the new topology $\mathcal{T}_{i+1}^j$ is used in the next iteration as $\mathcal{T}_{i+1}$. If the maximum is lower than $L(\mathcal{T}_i)$, the new topology is discarded and the original HMM will be regarded as the optimum topology.

Once model j is to be removed from the topology, the number of Gaussians in the remaining models is increased until the total number of Gaussians in the original topology and in the new topology are equal.

In the original algorithm, at each merging iteration, the topology of the system is kept fixed except for the two models that are being merged. Therefore when the BIC score is positive and the two separate models are replaced with the merged model, the topology does not only improve locally but also as a whole. In the SHoUT$_{DCM}$ approach it is not possible to look at a local improvement, because the number of Gaussians of all remaining models can change. Therefore the score of the entire topology is used for comparison. As the case for the local BIC comparison, when the number of Gaussians is kept constant, it is possible to compare the old topology $\mathcal{T}_i$ with the new topology $\mathcal{T}_{i+1}$ without the use of a scaling parameter.

In order to make a fair comparison between the Viterbi score of the original system $\mathcal{T}_i$ and the new HMM topology $\mathcal{T}_{i+1}$, the number of Gaussians of the new system have to be increased until it is the same as the number of Gaussians in the original system. The following procedure is proposed for this:

- The number of Gaussians in model j is determined ($n(j)$).
- For each model x the amount of data that was aligned to it in system $\mathcal{T}_i$ is stored ($N(x,i)$).
- After removing model j, for each model x the amount of data that is aligned to it in the new system $\mathcal{T}_{i+1}$ is stored ($N(x,i+1)$).
- For each model x, the number of Gaussians that will be *added* to the mix, $\Delta n(x)$, is calculated as follows:

$$\Delta n(x) = \frac{N(x,i+1) - N(x,i)}{N(j,i)/n(j)} \tag{5.1}$$

Each model that is assigned new Gaussians will obtain them by splitting its Gaussian with the highest weight after each training iteration. After all GMMs consist of the correct Gaussians, all models are re-trained and the new overall Viterbi score is calculated. This score is compared to the original score. If this score is higher than the original, a new cutting iteration will be started. Otherwise, the original topology will be restored and the new topology will be discarded.

As shown in figure 5.5, the actual merging decision is taken as late in the process as possible, after all models are retrained. A positive BIC score truly means that the *entire* system is improved by cutting away a model. In the SHoUT$_{D06}$ system, the choice for either stopping or merging two models is done before the rest of the system is retrained (see figure 5.2).

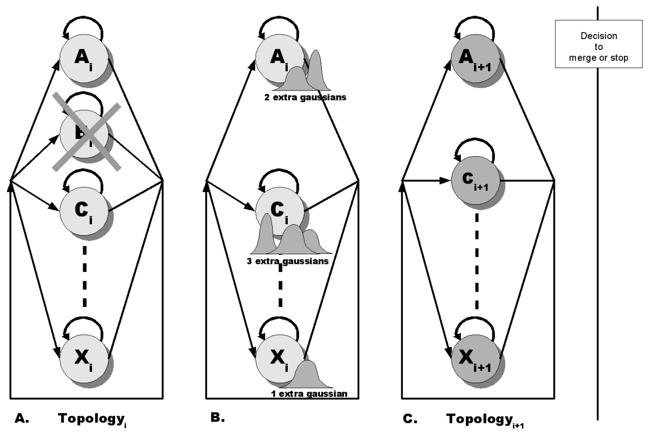


Figure 5.5: *In the SHoUT$_{DCM}$ system a single model is chosen to be removed from the topology (a). The number of Gaussians of the remaining models is increased so that the total number of Gaussians stays constant (b). Only after retraining all models it is determined if the new topology should be maintained (c).*

The complexity of the original SHoUT$_{D06}$ algorithm is $O(n^3)$ because of the pairwise comparison of each model at each time frame. The complexity of this algorithm is only $O(n^2)$ because per model only one possible new topology needs to be calculated at each time frame. Note that although theoratically this algorithm scales better to long recordings than SHoUT$_{D06}$, it is possible that calculating each new topology takes so much time compared to BIC that in practice the speed-up is limited.

RT06s evaluation

Not only SHoUT$_{D06}$, but also the SHoUT$_{DCM}$ diarization subsystem was evaluated at RT06s. The results of SHoUT$_{DCM}$ on the development set and on the evaluation data are compared with those of SHoUT$_{D06}$ in table 5.11. On the development set, the data from RT05s, SHoUT$_{DCM}$ outperforms SHoUT$_{D06}$. This is not reflected in the results of the RT06s evaluation where SHoUT$_{D06}$ performs better for the conference room data. As expected, the processing speed of the SHoUT$_{DCM}$ system (real-time factor 2.25) is better than the speed of SHoUT$_{D06}$ (4.63). These factors have been measured on an Intel Xeon 2.8 GHz processor.

Test set	SHoUT$_{D06}$ DER (%) no overlap	SHoUT$_{D06}$ DER (%) with overlap	SHoUT$_{DCM}$ DER (%) no overlap	SHoUT$_{DCM}$ DER (%) with overlap
RT05s conference room	21.6	30.2	18.6	27.6
RT06s conference room	22.7	37.2	25.2	39.5
RT06 lecture room	30.8	32.4	30.1	31.6
Processing speed (×RT)	4.63		2.25	

Table 5.11: *The speaker diarization results of the SHoUT$_{D06}$ and SHoUT$_{DCM}$ model-based systems measured with and without overlapping speech regions.*

Table 5.12 contains results of the two systems for each recording of the conference room RT06s evaluation set. For both systems the DER, average purity (see chapter 2, section 2.4.2) and the number of speakers in the hypothesis file are listed. For five of the eight recordings the DER of the two systems are similar, but the results differ considerably for three of the recordings. For the recordings EDI_20050216-1051 and EDI_20050218-0900, the SHoUT$_{DCM}$ results are poor and for VT_20051027-1400 the result of the SHoUT$_{D06}$ system is disappointing. For EDI_20050216-1051, the SHoUT$_{DCM}$ system cut away one of the reference speakers. This is reflected by the low average purity (68.4%) as this speaker is noise for models of the other speakers. In EDI_20050218-0900 the system did not over-cluster but instead did not merge enough models. In this case the purity is reasonably high compared to the SHoUT$_{D06}$ system (77.2% compared to 80.9%) but one of the reference speakers is still divided over two models. The SHoUT$_{D06}$ system made the same kind of error for the VT_20051027-1400 meeting. Although the average purity is comparable to the purity of the SHoUT$_{DCM}$ system, the final topology still contained too many models (8 compared to 6 models for SHoUT$_{DCM}$).

	SHoUT$_{D06}$			SHoUT$_{DCM}$		
Meeting ID	DER	Purity	Spk	DER	Purity	Spk
CMU_20050912-0900	20.71	82.9	7	24.53	82.6	6
CMU_20050914-0900	12.34	87.0	6	12.46	86.4	6
EDI_20050216-1051	20.97	82.0	5	**39.24**	**68.4**	4
EDI_20050218-0900	29.51	80.9	6	**44.89**	77.2	**7**
NIST_20051024-0930	8.90	86.9	5	10.61	84.7	4
NIST_20051102-1323	9.53	89.4	7	9.18	91.2	6
VT_20050623-1400	20.74	81.8	7	23.77	82.9	7
VT_20051027-1400	**52.18**	70.2	**8**	30.45	71.3	6

Table 5.12: *Results for the SHoUT$_{D06}$ and SHoUT$_{DCM}$ systems on the RT06s conference room evaluation data. For both systems the DER, average purity and number of hypothesis speakers are listed for each separate recording.*

Discussion

Based on the presented observations, it is hard to conclude which of the two systems is actually performing the best. None of the systems is outperforming the other systematically for all recordings. The purpose of dividing Gaussians over multiple models in the SHoUT$_{DCM}$ system was to create models with higher purity, but the average purity in table 5.12 is not consistently higher than for the SHoUT$_{D06}$ system. Also comparing the real-time factors for the two systems is a bit risky. Although as expected, the SHoUT$_{DCM}$ system is faster than the SHoUT$_{D06}$, it is questionable if this is really always due to the faster algorithm. In some cases the higher error rate of the SHoUT$_{DCM}$ system was caused by stopping merging too early. The better real-time factor of the SHoUT$_{DCM}$ system might be influenced by the gain in speed of using fewer merging iterations. Real-time factors are only really comparable when the diarization error rates are identical or at least similar.

One of the assumptions underlying SHoUT$_{DCM}$, is that merging pairwise decreases system performance when some of the models are impure. Therefore, for SHoUT$_{DCM}$ all Gaussians are spread over all remaining models instead of only increasing the number of Gaussians in the merged model. The analysis that was performed after RT06s, described in section 5.3, revealed that this assumption does not seem to be valid. As can be seen in table 5.6, the merging algorithm that merges models pairwise, contributes to the total DER marginally (4.17% absolute). Of course, errors also occur because of merging models from different speakers. The contribution to the DER because of these kinds of errors is a bit higher (10.37% absolute). In turn though, the SHoUT$_{DCM}$ system might be having problems with assigning the Gaussians to the correct remaining models. An analysis similar to the oracle-based analysis of SHoUT$_{D06}$ is needed to determine this, but unfortunately the SHoUT$_{DCM}$ system is not analyzed in-depth because of time constraints.

The SHoUT$_{DCM}$ subsystem seems less computationally expensive than the original SHoUT$_{D07}$ subsystem, but because of its different approach, it is impossible to tune SHoUT$_{DCM}$ in a way so that it can act with either the exact same accuracy as SHoUT$_{D07}$ (and same computational effort) or with a small degradation in accuracy but lower computational effort. The diarization subsystem that will be described in the following section, SHoUT$_{D07*}$, is able to switch between high accuracy and low computational requirements. This enables the system to handle short recordings with the accuracy of SHoUT$_{D07}$ while it is also able to process longer recordings.

5.5.2 The multiple merges system, SHoUT$_{D07*}$

The SHoUT$_{D07*}$ system is closely related to SHoUT$_{D07}$. Only a small modification has been implemented in order to speed up the process and to make processing of recordings up to two hours practically possible.

The number of merge calculations can be brought back by simply allowing merging of multiple models at a time. The two models A and B with the highest (and positive) BIC score are merged first. Next, the two models C and D with the second highest (and positive) score are merged. If this second merge involves one of the earlier merged models, for example model C is the same model as model A, also the other combination (B,D) must have a positive BIC score. This process can be repeated a number of times for each iteration as long as the BIC scores for all combinations of models in a merge is positive (see figure 5.6).

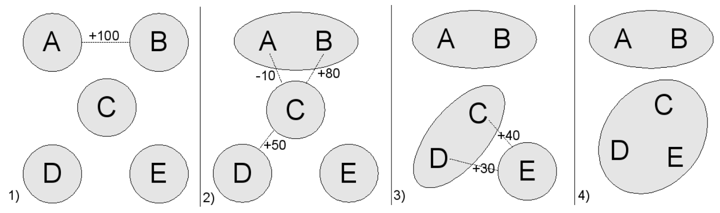


Figure 5.6: *Example of a situation where five models (first figure) are merged at a single diarization iteration. First in (2), the two models with the highest BIC score are merged. Then in (3), model C is not merged with {A,B} because the BIC score with A is negative. Instead it is merged with model D with the second highest BIC score. Finally model E is merged with {C,D} because both BIC scores are positive.*

If too many merges are allowed at a single iteration, the DER will increase. Increasing the number of merges at a time will decrease the number of Viterbi alignments and training iterations. As demonstrated in section 5.3, the system will not be able to create pure models if not enough alignments and training iterations are performed. On the other hand, restricting the number of merges at a single iteration to one will result in the original SHoUT$_{D07}$ system. Therefore, by varying the number of merges from one to many, the system can be very accurate but slow or very fast but inaccurate.

For the SHoUT$_{D07*}$ subsystem, the number of merges per iteration is not fixed, but dependent on the number of models in the topology as follows:

- If there are more than 20 models left in the topology, a maximum of four models is merged at a single iteration.
- If there are less than 20 models, but more than 10 models left in the topology, a maximum of two models is merged at a single iteration.
- If there are less than 10 models left in the topology, only 1 merge per iteration is allowed.

The SHoUT$_{D07*}$ is not yet tested at a Rich Transcription benchmark, but on the development set this procedure was used without performance loss with a reduction of the real-time factor from 8 times to 2.7 times real-time.

Note that although the SHoUT$_{D07*}$ algorithm was measured to be faster than the SHoUT$_{D07}$ algorithm, the complexity of the two algorithms is identical, $O(n^3)$. Therefore, for recordings longer than two hours, the SHoUT$_{D07*}$ algorithm will not solve the shortcommings of SHoUT$_{D07}$.

5.6 Conclusions and future work

In this chapter four versions of the SHoUT diarization subsystem were discussed. The initial version, SHoUT$_{D06}$, was improved considerably due to the comparison to the ICSI diarization system and thanks to a thorough oracle-based analysis. The findings of this analysis were incorporated into the second diarization subsystem, SHoUT$_{D07}$. This subsystem is able to perform state-of-the-art diarization without the use of prior trained speaker models. The final two versions of the diarization subsystem were implemented to make diarization of relatively long recordings feasible.

5.6.1 System analysis

The major part of the analysis performed in this chapter was based on oracle experiments in which part of the subsystem was replaced by the ground-truth. Two assumptions were done for these experiments that are debatable. First, it was assumed that the test sets used were large enough to produce statistical significantly results. The experiment results show high DER variations between individual recordings and therefore a high number of recordings is needed to 'smooth' this 'flakiness'. For this reason the test sets were chosen as big as possible. For the final experiments, 21 conference meeting recordings were used.

The second assumption that is debatable is that for the oracle experiments it was pretended that the components are mostly performing independent of each other. If the components act truly independently, it is possible to measure its performance by providing it perfect ground-truth input as was done in the oracle experiments. Unfortunately the performance of the components *is* most probably dependent on the performance of the other components. It would be better to analyze each component by providing input with various types of errors. Of course, such an analysis would be even more complex than the analysis performed in this chapter. Due to time constraints such an analysis has not been conducted.

Even though the test set was relatively small and no complex experiments were performed to test interaction between components, the analysis provided valuable information for improving the SHoUT$_{D06}$ diarization subsystem. The final analysis also provides valuable hints for future improvements. Especially the gap in performance of the system between the SDM and MDM tasks is striking. The systems used for the two tasks are identical except for the second feature stream that is applied for the MDM task. Therefore it seems likely that the SDM system can be improved considerably by introducing additional feature streams, similar to what was used in the MDM system. Of course these additional streams would not be based on delays since there is only a single microphone in the SDM condition, but other acoustic features could be used, or even the output of other speaker diarization systems as additional feature streams.

The final analysis described in section 5.4 also revealed to other interesting facts. It was shown that the SHoUT SAD subsystem described in the previous chapter improves the diarization performance considerably. Compared to the original SAD subsystem, the SHoUT SAD subsystem contains fewer false alarms. Especially false alarms decrease diarization performance as they act as noise during training of the speaker models. Secondly it was shown for the MDM condition that noise reduction improves diarization considerably.

5.6.2 Diarization for long recordings

The diarization subsystem is based on an agglomerative clustering approach. BIC is used for merging the clusters in such a way so that the overall complexity of the system remains constant and the λ factor does not need to be tuned on a development set but instead can be omitted (see chapter 2, section 2.4). The down-side of this approach is that comparing two models is computationally expensive and that the number of comparisons increases order two with the length of a recording. In order to solve this problem, the SHoUT$_{DCM}$ and SHoUT$_{D07*}$ diarization subsystems were developed.

Although the SHoUT$_{DCM}$ performed well at the RT06s benchmark, no further research was performed on the diarization method because for further development a thorough analysis such as performed for SHoUT$_{D06}$ would have been needed. Due to time constraints it was not possible to perform an analysis for both systems. Instead, the SHoUT$_{D07*}$ system was developed that is closely related to the SHoUT$_{D07}$ system. With the use of a single parameter, it is possible to decrease the real-time factor of this subsystem with a slight decrease in performance. For short recordings though, the SHoUT$_{D07*}$ system is identical to the SHoUT$_{D07}$ system.

It is interesting to aim future research for diarization of long recordings at the SHoUT$_{DCM}$ approach, but also at two other approaches. First, the process could be made faster by combining the SHoUT$_{DCM}$ approach and the SHoUT$_{D07}$ approach. For long recordings, the first iterations could be performed by SHoUT$_{DCM}$ while the final iterations are performed by SHoUT$_{D07}$. The experiments from section 5.5.1 indicate that SHoUT$_{DCM}$ seems weak in determining the optimal number of clusters, but it works fine in clustering the initial models.

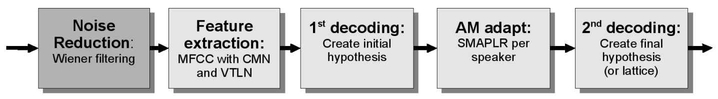
The second possible approach for diarization of long recordings is related to the approach taken for SAD in the previous chapter. Instead of processing the entire recording at once, it could be cut up in chunks and each chunk could be processed individually. Although for SAD it is relatively easy to re-combine the chunks, for diarization this step is not so straightforward. It is not that easy to determine which speaker model from one chunk matches the model of another chunk. If a method *is* found that can match the correct speaker models of the various chunks, it is possible to process recordings of infinite length. It would also make tracking of speakers over multiple recordings straightforward.

CHAPTER 6

AUTOMATIC SPEECH RECOGNITION

In the previous two chapters, segmentation and clustering have been discussed. These two subsystems are supposed to yield per speaker a cluster of speech fragments. The clusters are input to the subsystem that will be discussed in this chapter, the automatic speech recognition subsystem.[1]

As described in chapter 3 the SHoUT ASR subsystem creates a transcription in two *decoding* iterations. Anticipating the first iteration, feature extraction is performed and optionally a noise reduction filter is applied to decrease background noise. In the first iteration, a transcription containing timing information on phone basis, is generated that is used to apply unsupervised adaptation on the acoustic models for each speaker before running the final decoding iteration. Figure 6.1 summarizes this procedure.



Noise Reduction: Wiener filtering
Feature extraction: MFCC with CMN and VTLN
1st decoding: Create initial hypothesis
AM adapt: SMAPLR per speaker
2nd decoding: Create final hypothesis (or lattice)

Figure 6.1: *The two decoding iterations of the SHoUT ASR subsystem. The optional noise reduction is performed using the SRI Wiener filtering toolkit.*

This chapter concists of three parts: a part about modularity, about search space management and about robustness. Novel research was especially performed for the part about search space management. In the first part about modular design and the final part about robustness, important development choises will be discussed and the evaluation of these choises will be described.

Decoding plays a central role in each of the ASR iterations. In chapter 3 the requirements for decoding were defined and in this chapter the issues encountered during the development of the SHoUT decoder will be discussed. One of the most important requirements is that the decoder needs to be able to facilitate research

[1]The research presented in this chapter is, in part, published in [HOdJ07, HOdJ08]

into the ASR issues anticipated as relevant. It should be easy to test new ideas and techniques or to replace existing components with alternative implementations. For this to be possible, a modular design for the decoder is chosen with a straightforward interface between components. In section 6.1 the modular design of the decoder will be discussed.

Very important for every decoder for large vocabulary continuous speech recognition, and therefore also for the SHoUT decoder, is good management of its search space. In scenarios where the language to be processed comes with a large vocabulary, the possible number of combination of words that need to be regarded while searching for the correct string of words, can reach near infinite. A technique called Language Model Look-ahead (LMLA) can help the management system to better disregard the least promising paths, but on the architecture that is chosen for the SHoUT decoder, it is not straightforward to deploy full LMLA efficiently. In section 6.2 the research question: 'How can full language model look-ahead be applied for decoders with static pronunciation prefix trees?', is addressed and a method is proposed that makes it possible to deploy full LMLA for decoders such as SHoUT.

While it was possible to focus the research for segmentation and diarization directly on robustness against unknown audio conditions, for ASR it is first needed to solve the above mentioned problems before robustness techniques such as discussed in chapter 2 could be implemented. Due to time constraints, it was not possible to add all robustness techniques described in chapter 2, but a selection of known techniques was implemented for the SHoUT decoder. In section 6.3, where the research question: 'Which methods can be applied to make the decoder insensitive for a potential mismatch between training data and target audio?' is addressed, the performance of these techniques will be discussed.

6.1 Modular decoder design

The task of the decoder is to find the utterances that were most probably pronounced given the acoustic input (see chapter 2, section 2.1). In section 2.1.4 the algorithms for decoding based on Viterbi token passing were discussed in-depth. In this section, the design of the SHoUT decoder, which makes use of Viterbi token passing, will be discussed.

During the design phase of the SHoUT decoder, special attention was given to modularity. As described in chapter 2, a Viterbi decoder uses three knowledge sources for finding the most probable utterances: a language model, acoustic models and its pronunciation dictionary. Because these models are used simultaneously during decoding, it is tempting to design a decoder for which the code of these models is interweaved. Although such a design might result in a decoder optimized for speed, mixing code for these knowledge sources has two major disadvantages. First, it will decrease the readability of the source code, which can be a source of software bugs introduced during development and maintenance. Second, it will make it hard to switch to alternative implementations for the models as such a switch might affect code in every part of the decoder. A strict modular design will make it easier to replace Gaussian mixtures of the acoustic models by neural networks or to replace stochastic n-best language models by context-free grammars. This flexibility makes it possible to perform research on the various methods for modeling the acoustic and language knowledge.

In the remainder of this section, the design of the three types of models will be discussed followed by the architecture for the token-passing algorithm in which the three models have been integrated.

6.1.1 Language models

The decoder uses statistical N-gram backoff language models to determine its a-priori probabilities (see chapter 2, section 2.1.6). A number of tools exist to create N-gram language models out of text data collections. The most commonly used tool is the SRILM toolkit [Sto02]. This toolkit stores its models in standard ARPA format which will be the LM input file format for the SHoUT decoder. In the ARPA file format, each N-gram is stored in text on a separate line followed by its probability and backoff value. If the decoder would use this file format directly, it would need to compare the words of its hypothesis with the words of each N-gram until the correct N-gram is found. These kinds of text-based comparisons are computationally too intensive to be used for large language models. Therefore, a fast look-up method is needed for querying N-grams and their probabilities and backoff values.

Comparing numbers is a lot faster than comparing text strings. Therefore, each word in the dictionary is assigned a unique identification number $wordID$, and the words in each N-gram are replaced by these identification numbers. Even replacing each word with its $wordID$ takes up considerable computational resources when common string comparison is used to identify each word in each N-gram. Therefore, the words are first represented in a tree structure with a node for each character of the word and this tree is then used to find the correct $wordID$'s for each N-gram. Matching words with their $wordID$ by searching the tree is considerably faster than searching through the full list of words.

Even comparing sequences of numbers is time consuming. Therefore a method is needed to efficiently find the N-grams of $wordID$'s. For unigrams, this task is easy: all unigram probabilities and backoff values are stored in a list that is sorted on the $wordID$ of each unigram so that it is possible to obtain the probabilities of each $wordID$ with one single lookup.

For the higher order N-grams, it is not possible to use sorted lists that can be queried directly. Such lists would be multi-dimensional (three dimensional for tri-grams), very sparse and would take up too much memory. Instead, a *minimum perfect hash table* is used for querying higher order N-grams [CDGM02]. A minimum perfect hash table is a hash table where each slot in the table is filled with exactly one item. This means that it is possible to query N-gram probabilities in one single lookup and that no extra memory is needed except for storing the hash function and for the key of each data structure, the N-gram $wordID$'s. This key is needed because during lookup, the hash function will map queries for non-existing N-grams to random slots. By comparing the N-gram of the query to the N-gram of the found table slot, it can be determined if the search is successful. The algorithm proposed in [CHM92, CDGM02] is used to generate the hash functions.

The SHoUT decoder is able to handle language models up to 4-grams. It is possible to extend the decoder so that it is able to handle higher order N-grams, although the tables and the hash functions would then start to take up large amounts of memory.

6.1.2 Acoustic models

For acoustic modeling, standard three-state left-to-right hidden Markov models with Gaussian mixture models as probability distribution functions are applied (see chapter 2 and figure 2.1). This means that a training procedure is needed to set the HMM transition probabilities, GMM Gaussian weights, Gaussian mean vectors and covariance matrices. Also the number of Gaussians per mixture needs to be set and the triphone clusters need to be defined. In the SHoUT toolkit, this complex procedure has been implemented in a stand-alone application. The steps that the training application takes will be discussed next. Figure 6.2 is a graphical representation of the training procedure.

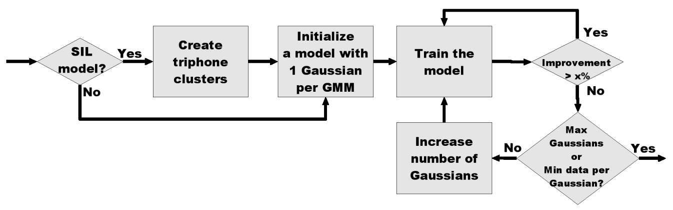

SIL model?
Yes
No
Create triphone clusters
Initialize a model with 1 Gaussian per GMM
Train the model
Yes
Improvement > x%
No
Increase number of Gaussians
No
Max Gaussians or Min data per Gaussian?
Yes

Figure 6.2: *The training procedure as it is implemented in the SHoUT toolkit. Each phone model is trained this way.*

Allocating training data to each phone model

In order to train the acoustic models, training data for each model is needed. The acoustic training data consists of the audio itself, the exact start and end time of each phone and of the neighboring phones (left and right context). The easiest way to obtain this information is to run a *forced-alignment* of each utterance in the training set. A forced-alignment is a decoding run with only one search path: the string of phones that make up the utterance. The timing information for each phone in the result of the forced-alignment can be used for training the models. Of course, this method requires that there are already acoustic models available.

Another method for creating a training set is to manually determine start and end time of each phone. This method is very time consuming. Fortunately, by annotating a relatively small set of utterances, initial models can be created that can be used for forced-alignment. The new (larger) training set can then be used to train new models and iteratively more refined models can be created.

A third approach is to map acoustic models from another language to the target language. Similar to the previous method, the initial forced-alignment will be rough, but by iterating the process, refined alignments can be created. This method was used to create the Dutch acoustic models for the SHoUT decoder. Publicly available English acoustic models were used as initial models and in each iteration the latest alignments were used to train new models. In total, eight iterations were run to create the final Dutch acoustic models.

Non-speech models

Once training data is allocated for each phone, the training procedure as depicted in figure 6.2 can be started. The SHoUT decoder uses two kinds of acoustic models: *phone models* with the described three state topology and *non-speech models*. The non-speech models represent not only silence, but also a variety of non-speech sounds such as lip-smack or laughter. They are not modeled with three states, but instead the HMMs of these models only contain one single state. The non-speech model training is not context dependent but instead all non-speech frames are mapped to the same GMM. Therefore, for the non-speech models it is not needed to determine the triphone clusters and only two transition probabilities need to be calculated. In all other aspects the training procedure for non-speech models is identical to that of the other models. Note that because the SAD subsystem is expected to remove all audible non-speech, the SHoUT ASR subsystem will only be trained with a non-speech model for silence.

Context dependency

Determining the triphone context clusters is done as proposed in [YOW94] and described in chapter 2. For each state, a decision tree is created in such a way that at each node, the training set is optimally divided in two. For each node in the tree a single Gaussian is trained on the data of that node. Note that this means that, not only the begin and end time of each phone occurrence should be annotated, but also the moments that a state transition is being made, so that each feature vector can be assigned to one of the states.

The size of each decision tree is restricted by three static parameters. First, a minimum number of training samples should be available for each node. Second, the improvement in score gained by splitting a node should be at least a fixed percentage and third, the depth of the tree is limited by a fixed number. Finding the optimum values for these three parameters by means of a grid search would have been too time consuming. Instead a number of explorative experiments was conducted to obtain rough settings for these parameters so that the number of clusters was balanced with the amount of available data for each cluster. The three parameters were finally set to a minimum of 2000 training samples per cluster, a maximum tree depth of 19 (resulting in maximal 2^{18} clusters) and a minimum score improvement of 50%.

Increasing the number of Gaussians

Once the triphone clusters are determined for each state, an initial model with one Gaussian for each cluster is trained. Two EM training methods are implemented for the training application: Baum-Welch and Viterbi. Using Baum-Welch training, all training samples are presented to all three states, but for all states the samples are weighted with the probability that they occur in that particular state. Note that in contrary to the determination of the triphone clusters, for this task the alignment of each phone occurrence is fixed but the moments of the state transitions are unknown (hidden). Using Viterbi training, each feature vector is presented only to the state for which it has the highest probability to occur in. With Baum-Welch training more accurate GMMs can be trained than with Viterbi training, but Baum-Welch is more time consuming.

In order to speed up the training procedure of increasing the number of Gaussians in each triphone cluster, Viterbi is used instead of Baum-Welch. Each triphone cluster is trained until the score does not improve more than a fixed percentage. As with determining the triphone clusters, determining the optimum value for this percentage is very time consuming. After explorative experiments it was set to 0.5%. After each set of training runs, the Gaussian with the highest weight (the Gaussian that was trained with the most training samples) is split in two. The two new Gaussians are created by shifting the means of the Gaussians to opposite sites in all dimensions[2] and by increasing the variance in each dimension by 20%. After splitting the Gaussian, each model is trained with another set of Viterbi training runs.

This procedure of increasing the number of Gaussians is repeated until a maximum number of Gaussians is reached or until the number of training samples for either of the Gaussians in the GMM reaches a minimum. Also this minimum is found by explorative experiments. It is set to 20 training samples per Gaussian.

Baum-Welch training

In an attempt to improve the precision of the models, after a number of iterations of increasing the number of Gaussians, instead of Viterbi training, Baum-Welch training is applied. Also, once the maximum number of Gaussians per GMM is reached, an extra number of Baum-Welch runs is performed. As the experiments in section 6.3.3 will show, these final Baum-Welch runs improve the models considerably.

6.1.3 Pronunciation prefix tree

The dictionary is transformed into a Pronunciation Prefix Tree (PPT) as described in chapter 2, section 2.1.7. The nodes in this tree represent phones as depicted in figure 2.3. Each node contains a phone ID number and also the left and right context of the phone, so that the correct context dependent acoustic model can be used during

[2]It is also possible to only shift the mean in the dimension where the variance is the highest as this is likely to be the dimension where it helps most to model the data with more than one Gaussian. No experiments where conducted to check if this approach would have been better

decoding. Each node is also able to store tokens for the token-passing algorithm (section 2.1.4).

The acoustic models are not actually copied into the PPT, but instead the PPT can apply the model its API function to calculate token values for each incoming feature vector. The API function is the only connection between the PPT and the acoustic models and this makes it easy to replace the models with other implementations (see figure 6.3).

6.1.4 Token-passing architecture

The SHoUT decoder applies the token-passing algorithm for calculating its hypotheses (chapter 2, section 2.1.4). As described in section 2.1.8, in order to incorporate language model information it is possible to either copy the PPT for each language context or to use one static PPT and store the context in token lists. The SHoUT decoder uses the latter method. At each leave of the PPT, the new a-priori probability is added to each token. Because tokens that do not have the same language model history (the same N-grams) should never be merged (section 2.1.8), each token is provided with a unique language model number. In the current implementation, this number is actually the pointer to the corresponding record of one of the hash-tables of the LM, but all that matters is that each N-gram has a unique number. The decoder only interfaces with the language model through its API when a new prior needs to be calculated. This procedure makes it possible to easily switch language model implementation.

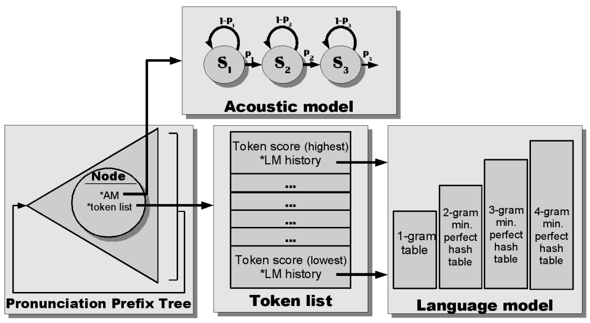


Figure 6.3: *The modular architecture of the SHoUT decoder. The acoustic model, language model and the pronunciation prefix tree are each implemented in their own module and communicate through a straightforward API. For the token passing administration, token lists are needed.*

In figure 6.3, the architecture of the SHoUT decoder is shown. It shows the three models and how they are connected by the token lists. With these four modules in place (AM, LM, PPT and token-list administration) the decoder is able to process audio using a small lexicon and language model. For large vocabulary speech recognition though, the number of needed token-lists and the number of tokens in each list will increase dramatically when processing each input feature vector. Large amounts of tokens will be a burden on both memory usage as computational effort and for large vocabulary recognition the real-time factor of the decoder will reach near infinity if no special measures are taken to restrict the number of tokens in the system. The measures taken by the SHoUT decoder in order to restrict the number of tokens and token-lists will be discussed in the next section.

6.2 Search space management

Decoding can be regarded as finding the optimal path in a large search space. When the token-passing algorithm is applied, each token represents a possible path in this search space. Especially with large lexicons and language models, the number of possible paths and therefore the number of tokens needed to run an exhaustive search can be dramatically high and the search can easily take up near infinite computational efforts. Therefore, it is very important that the search space is somehow managed and that the number of used tokens is restricted. The SHoUT decoder restricts the number of used tokens by disregarding less promising paths by deleting or *pruning* the corresponding tokens. This token pruning can be applied at a number of places in the decoder and in this section, the pruning methods applied by the SHoUT decoder will be discussed.

The risk of token pruning is that the optimal path, i.e. the token with the highest score after all feature vectors have been processed, could be deleted from the search space during one of the pruning runs. In order to prevent this, it is best to perform pruning based on both acoustical knowledge as well as language model information. Because in most decoders, the entire language model score is added to each token at the leaves of the tree, there will be a big gap between the scores of the tokens just before adding the LM priors and just after adding them. In combination with too aggresive token pruning, this gap might cause too many tokens to be pruned away. This problem can be solved by incorporating language model information at every node of the tree instead of only at the leave nodes. Language Model Look-Ahead (LMLA, [ONEC96]) makes it possible to incorporate approximations of the language model probabilities into the search tree at an earlier stage, before pruning is performed. This means that pruning is not only based on acoustical knowledge but also on linguistic knowledge. In [ONEC96] it has been shown that LMLA will allow for tighter pruning without loss of recognition accuracy.

For decoders that use copies of their Pronunciation Prefix Trees (PPT) to handle n-gram history, the LMLA probability approximations can be stored directly in the tree copies. Unfortunately, for decoders such as the SHoUT decoder that don't apply PPT copying but instead use a single static tree, it is not possible to store the LMLA probability approximations directly into the tree [ONEC96]. Without taking special measures, for these decoders the LM probabilities need to be looked-up at every time frame. The time needed to do so will diminish the advantage of the smaller search space. In this section a method will be provided that enables an efficient implementation of LMLA and reduces the LM probability look-up time for decoders that use static PPTs.

The remainder of this section is organized as follows. First, the token pruning methods applied by the SHoUT decoder will be discussed and in section 6.2.2 an overview of the use of LMLA in various systems is given and the method used by the SHoUT decoder is described. The section is concluded with experiments on the broadcast news domain (section 6.2.3) and with a discussion (section 6.2.4).

6.2.1 Token pruning

In token-based decoders, token pruning is applied to restrict the search space. Next, the commonly used token pruning methods will be described, followed by an extra token pruning method developed for the SHoUT decoder.

Common pruning methods

Two types of token pruning methods are commonly used in PPT-based decoders: *beam pruning* and *histogram pruning* [STN94]. With beam pruning, tokens with a probability value between the best found probability and the best probability minus a constant *beam* are retained at each time-frame. All tokens that are not within this *beam* are deleted. The SHoUT decoder uses two beam pruning methods. During *global beam pruning* all tokens of the entire PPT are compared to the best scoring token and pruned if necessary. *Word-end beam pruning* is done on all tokens that are at the leaves of the PPT and for which the LM probabilities are incorporated into their probability scores. This pruning method is used to limit the number of tokens that is fed back into the root node of the PPT.

Histogram pruning is also implemented in the SHoUT decoder. Here, only the best N tokens are retained when the number of tokens exceeds a maximum N which significantly restricts required memory [STN94]. This method is called histogram pruning because for sorting the tokens on their score, often a histogram is used [STN94]. Similar to beam pruning, histogram pruning is performed both globally (*global histogram pruning*) and in the leaves of the tree (*word-end histogram pruning*).

Pruning in SHoUT

Global pruning and word-end pruning, both by applying beam pruning or histogram pruning, are commonly used in PPT-based decoders. For decoders that use static trees such as the SHoUT decoder, a third pruning method can be used that is not explicitly mentioned in the literature. Instead of only pruning tokens that are in the word-end nodes, pruning is performed in each single node of the PPT. This pruning method, referred to as *single-state pruning*, restricts the length of each token-list. In the SHoUT decoder these token-lists are sorted on token score, which allows those lists to be pruned very efficiently (both for single-state beam pruning and single-state histogram pruning). Although the length of the token-lists does not influence processor load needed for calculating Gaussian Mixtures, it does take a lot of processing time to merge long lists and to calculate LM probabilities for all tokens. When LMLA is used, the lists need to be re-ordered in each compressed node (see below). Therefore restricting the length of the lists by using these two pruning methods is likely to speed up the search considerably. In section 6.2.3, the pruning methods are evaluated. Experiments will show that indeed single-state pruning reduces the processing time considerably.

6.2.2 Language model look-ahead

Language model knowledge is added to the hypothesis score at the PPT leaf nodes. Beam pruning is done earlier in the tree solely on the basis of acoustic evidence. Incorporating the LM model in an early stage into the tree will make it possible to compare and prune hypotheses on both linguistic and acoustic evidence. LMLA [ONEC96] achieves this by calculating for each token in the tree the LM probabilities of all words that are reachable from that token and temporarily adding the best one to the token's score. When the token reaches a leaf node, the temporary LM probability is replaced by the probability of the word represented by the leaf node. Following this procedure, narrower beams can be applied during pruning so that fewer tokens need to be processed and decoding is speed up considerably. On the down side, calculating all possible LM probabilities for all tokens takes a lot of time. In the literature a number of methods to manage these calculations is proposed. First, these solutions will be discussed and then the solution developed for the SHoUT decoder will be described.

LMLA in other systems

The least complex way for reducing the number of LM look-ups while applying LMLA is to use unigram probabilities for the look-ahead. By using unigrams the approximation of the best final LM score will be less precise, but it becomes possible to integrate these look-ahead scores directly in the PPT. In this case, each node stores a single value: the difference between the best LM score from before and after entering the particular node. Because only unigrams are used, these look-ahead values can be applied for all tokens, no matter their n-gram history. Unfortunately, it was shown that unigram look-ahead is outperformed by higher order look-ahead systems [ONEC96, CDGM02].

Another method for reducing the number of language model probability look-ups is proposed in [ONEC96]. All nodes of the PPT with only one successor node are skipped for calculating the LMLA values. The resulting compressed PPT will never require more nodes than twice the number of words from the PPT, reducing the number of needed LM lookups (see figure 6.4). The decoder in [ONEC96] uses tree copies in order to incorporate the LM probabilities. LMLA is performed on demand whenever a new copy is needed.

In [MNL05] at each node in the compressed PPT a list is stored with all words that are still reachable from that node. For small word lists, the look-ahead value is calculated exactly (each trigram probability is calculated and the best is chosen). Large word lists, at the root of the PPT, are skipped. For all other lists, the intersection with the n-gram lists are calculated before calculating the LMLA values. This saves a considerable amount of search time for those words that do not have a trigram

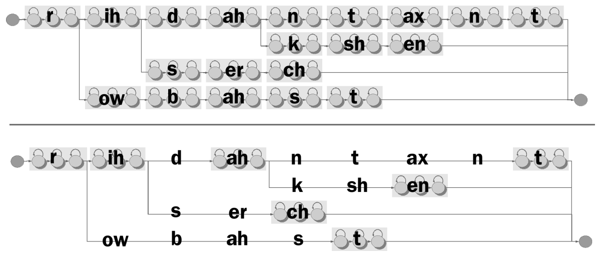


Figure 6.4: *Top: a pronunciation prefix tree with the four words: 'redundant', 'reduction', 'research' and 'robust'. Bottom: the same tree but now compressed for LMLA. Lookahead is only done in the remaining nodes.*

or bigram LM value.

Similar to the systems described in [CDGM02, SMFW02], the SHoUT decoder does not make tree copies. Instead, LM histories are stored in the tokens and the PTT is shared by all tokens. Therefore, Storing the LMLA values directly in the tree is not possible. To circumvent this problem, in [CDGM02] and [SMFW02] an LMLA cache is created in each node of the tree. These small caches contain LMLA values of earlier computed LM histories. Although the caches are highly optimized, the procedure takes more time than reading the single values directly when tree copying is applied. The LMLA data structure proposed below makes it possible to obtain a pre-calculated lookahead value in a static tree without searching in a cache.

LMLA in the SHoUT decoder

Figure 6.5 is a graphical representation of the data structure used to speed up language model look-ahead in the SHoUT decoder. Each node in the static PPT that has more than one successor or that is a leaf node (each node that is part of the compressed tree as described in [ONEC96]) is assigned a unique *LMLA index* value. LMLA probabilities are not stored directly in the nodes, but in *LMLA field* structures. Each structure contains the look-ahead values for tokens with one particular language model history in an array of probabilities P. The *LMLA index* of a node points to the corresponding LMLA probability in P. The probability array P is filled using the dynamic programming procedure described in [ONEC96]. Starting at the leaf nodes the LM probabilities are calculated. The probabilities are propagated backwards to the root of the tree and at each branch the maximum probability is selected and stored in P. Using this method, each candidate LM probability is calculated exactly once. Each LMLA field structure is stored in a global hash table.

When a new token enters the root of the PPT, the LMLA field structure with the identical LM history as the token is looked up in the hash table. If a field structure with the same LM history does not yet exist, a new one is created and added to the hash table. A pointer to the LMLA field structure is stored in the token. Once the token has been linked to a field, obtaining the LMLA probabilities is straightforward.

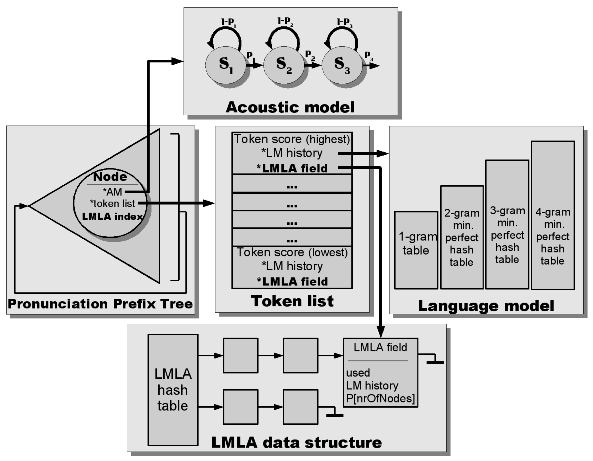


Figure 6.5: *The architecture of the SHoUT decoder including the module for language model lookahead (LMLA). Also an LMLA-index is added to each node in the PPT and a pointer to an LMLA field is added to each token list (marked in bold text).*

When a token is propagated to a new node, it uses this node's LMLA index on the probability array of the LMLA field structure. This action only takes two look-ups: retrieving the LMLA index and retrieving the probability. The search through the hash table is needed only once. After that, the time needed for look-ups is negligible.

The boolean parameter *used* of the LMLA field structure is set to true each time the field object is used for a look-up. All LMLA fields that are not used (*used* parameter is false) during a fixed time window are deleted in order to save memory.

6.2.3 Experiments

First, the efficiency of the pruning methods and LMLA is tested in three experiments on the broadcast news domain. Next, two additional experiments are conducted in order to determine if unigram LMLA is equally effective as using n-gram LMLA. All experiments are conducted on Dutch broadcast news recordings. The first set of experiments are performed using our broadcast news development set. The second set of experiments is conducted on the recently made available development set of N-Best [KvL07], the decoding benchmark for Dutch (see appendix A).

Pruning and LMLA

Three experiments have been conducted to test the pruning methods and the LMLA efficiency. For the first experiment, only global beam pruning, global histogram pruning and word-end beam pruning was used. In the second experiment, state beam pruning and state histogram pruning were used as well. Finally, in the third experiment, also LMLA was applied. In order to ensure that it is possible to compare the experiments purely on basis of their needed processing time, the pruning configuration of each experiment was chosen in such a way that the word error rate is the same for all experiments (29.5%). When the WER is not equal for each experiment, it is impossible to conclude if differences in speed are due to improvements in the system or simply because too many paths in the search space are disregarded (resulting in a faster system but in higher error rates).

The pruning values for the three experiments shown in table 6.1 are obtained by first setting the global beam threshold so that the WER is 29.5% and then setting the other thresholds as tight as possible without influencing the word error rate.

	global	word-end	state	
ID	beam, hist.	beam	beam, hist.	LMLA
exp-1	210, 125000	45	no, no	no
exp-2	210, 125000	45	75, 250	no
exp-3	150, 40000	45	50, 250	yes

Table 6.1: *The pruning configuration of each experiment. The first number in the global end state columns represent the beam value. The second number is the histogram value.*

For decoding the broadcast news recordings, a dictionary containing 65K words was used. At the time of these experiments, the decoder was not yet able to handle 4-grams and therefore a trigram LM was used. All experiments were conducted on a machine with a 3.4 GHz Intel Xeon processor. For each experiment the number of search errors [Cha97], the WER and the real-time factor of the system have been calculated. Finally, also the average and maximum number of active nodes and number of tokens needed to decode each sentence were stored. For the third experiment also the maximum and average number of LMLA lookup tables were stored.

Results

For all three experiments, the word error rate is 29.5% and 3.5% WER is due to search errors, showing that the system performance is the same for each experiment. The real-time factor (RTF) of the first experiment (only global and word-end pruning) is 27.4 while the RTF of the third experiment is 10.5. This improvement is obtained because of the drastic reduction of active nodes and tokens due to LMLA. In table 6.2 all measured statistics of the three experiments are listed. The unused fields of the LMLA hash table are not deleted every time-frame (10ms) but every 25 frames. This means that the actual average *active* fields is less than the 51 mentioned in table 6.2. The number of tokens is measured each time-frame before histogram pruning. Therefore it is possible to have an average number of tokens that is higher than the histogram pruning threshold.

As can be seen from table 6.2, the second system (with state pruning) is roughly 9% faster than the first system (without state pruning) while the average number of tokens is only reduced by 1.5%. This is explained with the optimization that is possible when single-state pruning is applied: single-state pruning can be applied directly during the merging of two token-lists coming from different HMM states into

		Average number of		
ID	RTF	nodes	tokens	LMLA fields
exp-1	27.4	43314	126782	n.a.
exp-2	24.9	43730	124839	n.a.
exp-3	10.5	11395	20686	51

Table 6.2: *The measured statistics of the three experiments. For all experiments the WER is* 29.5%*.*

the same state. Without single-state pruning, all tokens of both lists need to be placed into the merged list, whereas when using single-state pruning, the merged list is finished as soon as the maximum number of tokens is reached (because all token-lists are sorted by score).

Unigram LMLA

As discussed before, one solution for reducing the needed number of LM look-ups is to apply unigram LMLA instead of full n-gram look-ahead. In order to prove that this method is not as efficient as full LMLA, two extra experiments were conducted. First the system with full LMLA is run and after that a system that only uses unigram LMLA is evaluated. As before, for these two experiments the pruning parameters are determined in such a way that the WER is equal for both experiments (29.6%) and the performance can be measured by the real-time factor. The experiments are performed on the development set of the N-Best task. In table 6.3 the results of the two experiments are listed. Although in the second experiment, less time was spent in querying LM n-grams, more tokens and token-lists were needed. These extra tokens slow down the decoder considerably.

LMLA method	global beam, hist.	word-end beam	state beam, hist.	RTF
Trigram	150, 40000	50	65, 160	14.0
Unigram	160, 85000	50	75, 210	18.9

Table 6.3: *The settings used for the unigram and trigram LMLA experiments. To obtain a WER of* 29.6%*, for unigram LMLA, wider pruning settings are needed resulting in an increase of the real-time factor of* 35%*.*

6.2.4 Discussion

The experiments discussed in this section show that the measures taken to manage the search space of the SHoUT decoder are effective and that it is possible to use the decoder for large vocabulary tasks. The single-state pruning method reduced the real-time factor of the system considerably and also the architecture for performing LMLA efficiently in static tree-based decoders helps increasing the decoder speed performance. It has also been shown that in the SHoUT decoder, this LMLA architecture outperforms unigram look-ahead.

Although the unigram LMLA system was 35% slower than the full LMLA system, it must be noted that the system without LMLA was even 2.4 times as slow as the optimal system. The fact that unigram LMLA already provides a considerable speed-up, and that it is less complex to implement than full LMLA, could be a consideration to chose for unigram LMLA. Also, note that the reported real-time factor results are closely related to the implementation of the SHoUT decoder and that it is possible that, if implemented in another decoder, the RTF gain of the full LMLA system compared to the unigram LMLA system is less distinct. Given this caveat, the experiments with the SHoUT decoder are very promising and it is our believe that full LMLA using the proposed data architecture will also improve the real time factor of other token passing decoders.

Some decoders use pre-compiled caches for LM probability look-up of the most occurring words. This helps because these words are used considerably more often than the remaining words and therefore have a high probability of being looked up. In the SHoUT decoder, no cache is being used, but instead a very efficient LM look-up method (discussed in section 6.1.1) is implemented that reduces a regular n-gram query to calculating the key for a minimum perfect hash table and using this key to directly access the probability. A cache might be useful for speeding up the calculation of the key, but the effect of this speed-up will be highly limited.

The techniques discussed in this section are needed for the decoding of audio when deploying large vocabularies and large language models. Numerous other techniques could be implemented to even speed up the decoder further, but given that the real-time performance is good enough for practical usage, the development goal is reached.

6.3 Robust ASR

For decades, research has been performed on making automatic speech recognition more robust in various kinds of ways. For example, numerous methods have been developed for noise reduction, for robust feature extraction in noisy environments or for creating more uniform acoustic models. These methods all aim at the creation of systems that are insensitive to the potential mismatch between training data and target audio. It was not feasible to implement all of these methods and subject them to experimentation. Only a selection of these methods could be chosen, and as described in chapter 3, the decision of which methods to pick was mainly guided by experiences reported by research groups participating in benchmark tests.

Figure 6.1 provides on overview of the possible steps in the SHoUT decoder that can be taken to perform robust ASR. This figure shows that before feature extraction, Wiener filtering can be applied for noise reduction. Note that for this, the ICSI toolkit is used. After Wiener filtering, Common MFCC feature extraction is performed. The Direct Current (DC) offset of the amplitude signal is normalized by subtracting the mean of the signal from each sample and a pre-emphasis filter is used that amplifies the high frequencies of the signal. For each speaker in the recording the Vocal Tract Length (VTL) warping factor is determined. This factor is used to scale the Mel-scale windows. This procedure is described in section 6.3.1. Finally Cepstrum Mean

Normalization (CMN) is performed.

After the first decoding run (not counting the run for determining warping factors), it is possible to apply unsupervised adaptation to the acoustic models before running the second decoding iteration. As adaptation method, the Structured Maximum a Posteriori Linear Regression (SMAPLR) method developed by [SML00] has been chosen because this adaptation method can be used on data sets of varying sizes without tuning any parameters (see chapter 2). As stated before, it is regarded an advantage when no parameters need to be re-tuned when conditions change. SMAPLR can be applied with maximum efficiency when only a low amount of adaptation data is available (a minute or less) or when several minutes or hours of data is available.

For evaluation of the ASR subsystem a number of Dutch acoustic models are trained. In the remainder of this section the steps taken to train these models will be discussed and the section will be concluded with the evaluation of the various methods.

6.3.1 Vocal tract length normalization

Any variation of vocal tract length between speakers results in less distinct optimal values of the acoustic model parameters. If no normalization is applied to reduce this variation, more Gaussians are needed to properly model a phone than if normalization *is* applied. A very effective method for normalizing for this variation was proposed by [CKA95] and discussed in chapter 2, section 2.5.1. The SHoUT decoder applies this normalization method where the feature vectors are obtained after shifting the Mel-scale windows by a certain warping factor. If the warping factor is smaller than one, the windows will be stretched and if the factor is bigger than one, the windows will be compressed. To normalize for the vocal tract length, large warping factors should be applied for people with a low voice and small warping factors for people with a high voice (note that this is because of how the normalization method is implemented. In the literature often big warping factors are linked to high voices instead of low voices).

In order to determine the speakers warping factors, both for during training the phone models as for during decoding, the method proposed by [WMOP96] is used (see section 2.5.1). A Gaussian mixture model (referred to as VTLN-GMM) is trained with data from the Spoken Dutch Corpus (CGN, see appendix A). In total, speech of 200 male speakers and 200 female speakers is used. The GMM only consists of four Gaussians. For the 400 speakers in the training set, the warping factor is determined by calculating the feature vectors with a warping factor varying from 0.8 to 1.2 in step sizes of 0.04 and determining for each of these feature sets what the likelihood on the VTLN-GMM is. For each speaker the warping factor used to create the set of features with the highest likelihood is chosen.

After each speaker is assigned a warping factor, a new VTLN-GMM is trained using normalized feature vectors and again the warping factors are determined by looking at a range of factors and choosing the one with the highest score. This procedure is repeated a number of times so that a VTL normalized speech model is created. From this point on, the warping factor for each speaker is determined by looking at a range of factors on this normalized model. Note that this method is very quick and that it is no longer needed to run an initial decoding run in order to obtain the warping factors. This is especially important for situations where it is not yet possible to obtain low word error rates, because in these situations the initial hypothesis might contain so many errors that it is hard to determine the correct warping factor on basis of this hypothesis.

Figure 6.6 contains the warping factors of the hundred speakers of the VTLN-GMM training set split in male and female. Three training iterations are performed. At the third iteration a model with eight Gaussians is trained. This model is picked to be the final VTLN-GMM.

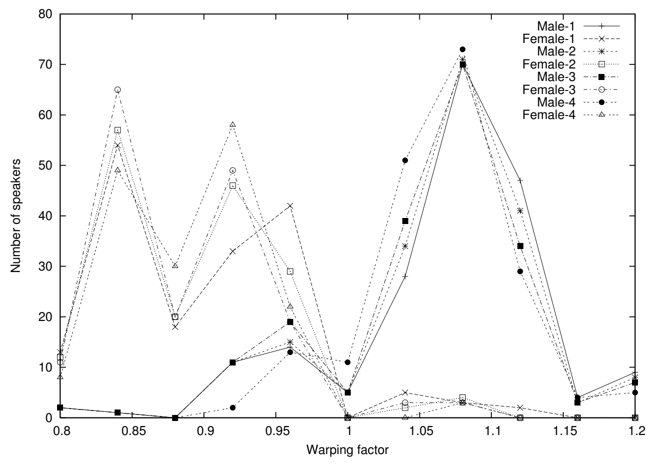


Figure 6.6: *The warping factor of* 200 *male and* 200 *female voices determined using the VTLN-GMM after each of its four training iterations. The female speakers are clustered at the left and the male speakers at the right.*

As can be seen in figure 6.6, the warping factor for male speakers is generally high and for female speakers the average warping factor is less than one. For the male speakers a distinct peak in the graph is visible. For female speakers though, figure 6.6 contains two peaks. It is unknown why this is the case and the graph does not contain a single peak for the female speakers as well. It is surmised that there might be two peaks because the model is created with a relatively small amount of data. Perhaps the graph would have a Gaussian form when a high number of speakers is used. Note that even with the small amount of data, the dip in the graph is reduced a little after each iteration.

6.3.2 Acoustic model adaptation

In chapter 2 the Structured Maximum a Posteriori Linear Regression adaptation method (SMAPLR) has been described [SML00]. This adaptation method applies a binary tree in order to determine the clusters of triphones of which the Gaussian mean vectors will be adapted. Each leave node of the tree represents a cluster and the more adaptation data is available, the more nodes the tree will have. MAP adaptation is performed at each leave and the prior matrices that are needed during this calculation are provided by the parent of each node. At each node, MAP adaptation is performed and the transformation matrices are sent down the tree to function as prior matrices. This method ensures that when only little data is available, the acoustic model will not be over-adapted, but when large quantities of data is available, this data will be optimally used to adapt the acoustic model.

The SMAPLR adaptation method uses a number of system parameters that need to be tuned. This is done once on a small part of the development set. It is believed that these parameters are not sensitive for changes in domain or audio conditions and do not need to be re-tuned when encountering new data.

The first parameter used by SMAPLR is the C-factor. This factor weights the prior matrices. The lower the C-factor, the more the adaptation depends on the prior matrices. In [SML00] it is shown that the adaptation works best when the C-factor is low at the root of the tree and higher at its leaves. The second parameter is the amount of adaptation data that should minimally be available for each node to perform MAP adaptation on. If this amount is too low, the system will over-adapt the acoustic model, but when it is too high, not enough clusters will be generated in order to optimally adapt the models. The third parameter that is not mentioned in [SML00], but that is added for practical reasons is the maximum depth that the tree is allowed to be. This parameter is added so that adaptation with large amounts of data (for example for speaker adaptive training) will not take up too much system memory.

For the SHoUT adaptation application, after explorative experiments, the minimum amount of required data per node was set to 2000 feature vectors (weighted) and the maximum tree depth was set 14 . With these settings, reasonable large trees will be created when high amounts of data are available and only a small tree will be used when small amounts of data are available. In order to test the best setting for the C-factor, three sets of experiments are conducted on one single speaker. Supervised adaptation is performed on the speech of this speaker, using varying amounts of adaptation data and three different settings for the C-factor. As can be seen in table 6.4 from [vV07], although differences are marginally, the best setting is 0.01 which is the same result obtained in [SML00]. As expected, the results using a small C-factor are poor when large quantities of adaptation data are available because the strong prior information is constraining the transformation estimations too much. On the other hand, when the prior is restricted (C=0.1), the models might be over-adapted when only a few utterances are available. Two remarks need to be made when the results of table 6.4 are compared to [SML00]. First, it is unclear what the average length of a sentence is in [SML00] and therefore it is hard to compare the results on basis of the number of sentences. Second, the method proposed in [SML00] to increase the C-factor as the tree becomes deeper was applied for the experiments listed in table 6.4. This means that for large quantities of data, at the leaves of the tree the C-factor is 1.0 for all experiments. This explains why the C-factor influences the results less than in [SML00].

Number of utterances	C=0.001 %WER	C=0.01 %WER	C=0.1 %WER
3	29.7	28.7	28.7
20	27.7	26.9	29.2
50	27.6	27.0	27.6
100	27.3	25.9	25.8
300	26.2	25.1	25.3

Table 6.4: *Supervised adaptation results of a single speaker with varying amounts of adaptation data and three settings for the C-factor. The WER without adaptation was* 29.2%

6.3.3 Evaluation of the robustness techniques

In this section a number of robustness techniques implemented in the SHoUT decoder has been presented. In the remainder of this section, a number of experiments that are conducted to test the efficiency of these techniques will be described. These experiments are conducted on a broadcast news test set with manually annotated speaker and segment boundaries. The total length of the test set is 84 minutes and it contains 114 speakers (see appendix A, section A.2). Experiments on other domains and with automatically obtained speaker clusters will be discussed in the next chapter (section 7.1). The experiments described now are set up solely to test the decoder and not to test interaction with the segmentation and clustering systems. First, a baseline experiment is described in which non of the techniques is applied. Then, one at a time, the techniques are added to the decoder. It is expected that the word error rate will decrease for each experiment.

CMN and VTLN

First, acoustic models are trained to check the efficiency of cepstrum mean normalization and vocal tract length normalization. For this, three models are created. The first model does not make use of either CMN or VTLN. For the second model, CMN is applied and the final model uses both CMN and VTLN. These models are trained only up to sixteen Gaussians per mixture in order to reduce the time needed to create the models. Both CMN and VTLN normalization are performed using clustered data of each speaker. In table 6.5 the results of the three experiments are listed. As can be seen, both CMN as VTLN aid in reducing the word error rate. Unfortunately, with only sixteen Gaussians and a relatively small test set it was not possible to prove that the combination of CMN and VTLN improves the recognition significantly. Even though, because the results of CMN and VTLN combined are promising it is chosen to proceed with both normalization methods in training models with more parameters.

AM type	%WER	Significance (p)
Baseline	33.2	
CMN	30.9	< 0.001 (compared to Baseline)
CMN and VTLN	30.5	0.327 (compared to CMN)

Table 6.5: *Evaluation of ASR on the Dutch BN test set using no normalization technique (baseline), CMN or CMN and VTLN combined.*

Number of Gaussians

In the experiments listed in table 6.6 CMN and VTLN are both applied and the number of Gaussians of the acoustic model is increased up to sixty. As can be seen in this table, the more Gaussians are used, the better the model is performing. Adding even more than sixty Gaussians would have limited impact though because the number of Gaussians is also limited by the minimum needed amount of data to train each Gaussian of each triphone cluster. As the amount of training data is limited to approximately 100 hours of speech, most phone models reach this ceiling before the sixty Gaussians are reached. The matched-pair significance test [PFF90] shows that the improvements are significant with $p < 0.03$.

Number of Gaussians	%WER	Significance (p)
16	30.5	
30	29.3	< 0.001 (compared to 16 Gaussians)
60	28.7	0.029 (compared to 30 Gaussians)

Table 6.6: *Evaluation of ASR on the Dutch BN test set using acoustic models with increasing maximum number of Gaussians per GMM.*

Baum-Welch training

As described in section 6.1.2, training the acoustic models with Baum-Welch instead of Viterbi is more time consuming but leads to more accurate models. Therefore the models are trained with Viterbi and every few iterations a Baum-Welch run is performed. In this experiment, the final model with maximal 60 Gaussians per GMM is trained another fifteen iterations with the Baum-Welch procedure. The results are listed in table 6.7. It shows that the model is refined and the WER is improved with 3.3% relative, but it could not be shown that the improvement is significant.

Segment normalization or speaker normalization

For both CMN and VTLN it is possible normalize each separate speech segment or to normalize each individual speaker cluster. In table 6.8 the results are listed of four

Acoustic model	%WER	Significance (p)
CMN, VTLN, 60 Gaussians	28.7	
CMN, VTLN, 60 Gaussians, extra Baum-Welch	28.4	0.097

Table 6.7: *Evaluation of ASR on the Dutch BN test set using one AM without and one AM with* 15 *extra Baum-Welch training iterations.*

experiments in which CMN and VTLN are performed on either segments or speaker clusters. As can be seen in table 6.8, the result is the best when both normalization methods are performed on basis of speaker clusters. The matched pair significance test shows that the system that applies both CMN and VTLN on a cluster basis, is significantly better than the systems that apply VTLN on a segment basis (both systems with $p = 0.03$), but for the other combinations of systems it could not be proven that there is a significant difference.

	%WER VTLN per segment	%WER VTLN per cluster
CMN per segment	28.9	28.7
CMN per cluster	28.9	28.4

Table 6.8: *CMN and VTLN normalization based on speech segments or speaker clusters.*

Noise reduction

The ICSI toolkit is used with default settings to reduce the noise in the audio recording of the Dutch broadcast news test set (see appendix A, section A.2). This tool applies a Wiener filter on the audio file and returns a noise-reduced file. This file is used as input for another decoding experiment. The results of this experiment are listed in table 6.9.

Noise reduction	%WER	Significance (p)
No noise reduction	28.4	
Wiener filtering	28.5	0.303

Table 6.9: *Evaluation of ASR on the Dutch BN test set with and without noise reduction.*

The word error rate with noise reduction is a bit higher than without noise reduction (0.1% absolute). It is not proven that the difference is significant though. The results do not show that noise reduction hurts the recognition, but apparently for broadcast news recordings, it does not improve the system significantly either. Therefore, for the broadcast news system it is decided not to use Wiener filtering as most of the audio does not contain any noise.

Unsupervised adaptation

Unsupervised acoustic adaptation is performed with SMAPLR as described in chapter 2 and section 6.3.2. The acoustic model is adapted for each speaker to create speaker dependent models. As can be seen in table 6.10, adaptation improves the word error rate with 4.2% relative. A second iteration of adaptation does not improve the result, but instead significantly degrades the performance.

Acoustic model	%WER	Significance (p)
First decoding iteration	28.4	
Second decoding iteration	27.2	< 0.001 (first iteration)
Third decoding iteration	27.4	< 0.001 (second iteration)

Table 6.10: *Evaluation of ASR on the Dutch BN test set using unsupervised SMAPLR acoustic adaptation.*

6.3.4 Discussion

In this section the measures taken to make the decoding process as robust as possible were discussed. The methods used either try to improve the input signal for decoding (CMN, Wiener filtering) or try to reduce variability in the training and evaluation data (VTLN, SMAPLR). Conform the literature, for the broadcast news domain the experiments show that these methods improve the result.

Numerous other normalization techniques for robust ASR are available and worthwhile investigating. For example, normalizing the decoder for speech rate has been shown to improve recognition results in spontaneous speech [Met05]. Speaker Adaptive Training (SAT) for example, improves recognition after (unsupervised) adaptation because all training speakers are normalized using their own adaptation matrices [AMSM96]. Similar to VTLN, histogram normalization, where the Mel-windows of each speaker are normalized, proved to be efficient for normalizing the feature vectors [Mol03]. Unfortunately these techniques could not be implemented and tested within the time frame of this research. Fortunately, the modular design of the decoder (see section 6.1) makes it possible to experiment with new or existing techniques in a straightforward matter, and in the future expectedly more normalization techniques will be incorporated in the SHoUT decoder. Currently, histogram normalization and speaker adaptive training are being implemented.

6.4 Conclusions and future work

In this chapter the SHoUT ASR subsystem was discussed. The most important design goal for this system was that, although the decoder needs to perform state-of-the-art, it needs to be possible to perform ASR related research using this system. With the modular design of the decoder, described in 6.1, this goal was reached. A number of ASR related research is already being performed using the SHoUT toolkit. For example, both the segmentation as the diarization software have been built on top of the decoder software described in this chapter. These applications make use of the HMM and the token passing modules while the language model is omitted and the structure of the lexical tree is modified. Also the software proved to be easily extended with the extra module needed for the first research topic using the SHoUT decoder: efficiently managing the decoder's search space (see figure 6.5).

Section 6.2 discussed the methods applied to restrict the search space of the SHoUT decoder. Common beam pruning, as well as pruning within single token lists (single-state pruning) and a method to efficiently use language model look-ahead in the SHoUT decoder or in any other decoder using a single pronunciation prefix tree were discussed and evaluated. In this section it was shown that single state pruning and LMLA speed up the decoder considerably without loss of recognition precision.

Finally in section 6.3 the implementation of a number of known techniques to perform more robust ASR was described and evaluated. The experiments show that these techniques are effectively implemented in the SHoUT decoder. Special care is put in selecting the various methods. For VTLN, a method was used that only requires a single GMM to determine warping factors instead of an initial hypothesis. In cases where the word error rate can be expected to be high (bad quality audio or speech), a bad initial hypothesis might negatively influence the correct picking of the warping factors. For acoustic model adaptation, the SMAPLR method was chosen because this adaptation method works optimally for both small amounts of adaptation data as for large amounts. As no parameters need to be tuned on development data (except for some system parameters that were tuned only once and do not need re-tuning when audio conditions change), this adaptation method can be safely applied for unknown audio conditions.

The SHoUT decoder has proven suitable for performing research on various ASR topics and in future work this decoder will be used again and its functionality will be extended. Initial progress is already being made in investigating Speaker Adaptive Training [AMSM96] or histogram normalization [Mol03]. In addition the decoder is being prepared to handle a second feature stream so that it is possible to use articulatory information as proposed in [Met05]. These methods are all good examples of techniques that improve the decoder's performance, but more important, that can aid in developing a decoder that is able to recognize speech in unknown audio conditions.

CHAPTER 7

SYSTEM EVALUATION

In the previous chapters the three subsystems that make up the SHoUT large vocabulary continuous speech recognition system have been discussed. For each subsystem a number of evaluations were conducted to ensure that the proposed techniques are performing adequately. The ASR subsystem was tested on a development set and the segmentation and diarization subsystems were tested at the NIST Rich Transcription 2007 (RT07) Meeting Recognition evaluation. In this chapter, instead of testing the individual subsystems, the entire system will be evaluated.

In this chapter, two questions will be addressed: 'How does the SHoUT system perform on the standard BN task?', and: 'How does the system perform on a surprise-data task with unknown audio conditions?'. In order to answer these questions, the system is evaluated on two tasks. First, the system is evaluated in the context of N-Best, the first Dutch ASR benchmark organized by TNO [KvL07]. This benchmark consists of tasks on multiple conditions and dialects. One of these tasks is the processing of Dutch broadcast news. This task is used to evaluate if the three subsystems perform well together as a single system. In the second task, the system is evaluated on the 2007 TRECVID evaluation collection (see appendix A). This collection consist of typical surprise data: audio recordings with unkown topics and with unknown audio conditions. It contains audio with varying topics (from poetry to children shows or broadcast news) and with a wide range of audio quality (recording conditions, background noise, etc). The results of the TRECVID evaluation will show if the robustness against unknown audio conditions of the SAD and diarization subsystems are helpful when the subsystems are employed in the SHoUT system.

7.1 N-Best

In 2006 a research project called N-best[1] was started. N-Best aims at setting up the infrastructure for a benchmark evaluation in large vocabulary speech recognition for the Dutch language, and at conducting the evaluation. This evaluation, the first one

[1]N-Best: Northern and Southern Dutch Benchmark Evaluation of Speech recognition Technology

for Dutch ASR, focuses on two tasks: broadcast news and Conversational Telephone Speech (CTS). Within these tasks, both Northern and Southern Dutch as it is spoken in the Netherlands and Flanders (Belgium) respectively, are evaluated.

In this section the SHoUT submission for this benchmark will be discussed. For this evaluation the SHoUT system as it was described in the previous chapters is applied with only a few minor adjustments. Next, before discussing the benchmark results and performing a post-evaluation analysis, these adjustments will be discussed.

7.1.1 System description

For the BN task, segmentation and speaker diarization are performed first, followed by the determination of the VTLN warping factor of each speaker cluster. The first decoding iteration is performed using BN acoustic models, vocabulary and language model. After unsupervised adaptation of the acoustic models of each speaker cluster, the final decoding iteration is performed.

For the conversational telephone speech, diarization is not needed because each speaker is recorded on his own channel. Some recordings in the development set contain cross-talk, the phenomenon that a speaker is recorded on the channel of the other speaker. Speaker diarization could have been employed to remove the cross-talk, but the N-Best project guaranteed that there will be no cross-talk in the evaluation data and therefore the diarization step is omitted. Instead, for each audio channel a simple energy-based segmentation is conducted in order to obtain the speech segments. These segments are used directly for determining the VTLN warping factor of each speaker. After a first decoding pass the acoustic models are adapted and with the second decoding run the final hypothesis is generated.

The data that can be used to train the statistical models is defined by the N-Best organization. The models discussed in the previous chapter are created with the Dutch BN part of this data set. In this section a number of experiments will be described in which the Dutch and Flemish data is mixed in order to create better models.

The BN and CTS systems both make use of a post-processing component at the very end of the process. This post-processing is needed because the scoring methods of N-Best are more strict than the methods used in the earlier experiments of this thesis. After discussing the training of the models, the post-processing step will be described.

The language models

The language model used for experiments in the previous chapter is based on Dutch newspaper text, but also on subtitles of BN shows. For N-Best only the newspaper data and transcriptions of the audio data are allowed to be used and therefore new language models need to be created. One language model is trained on Flemish data and the other on Dutch data. These two models are used for both the CTS and the BN tasks.

Table 7.1 contains the results of experiments on the Dutch BN development set (see appendix A) with the Dutch AM described in the previous chapter. The results show that the mix of newspapers and speech transcriptions performs best on this set. This language model is obtained by mixing the two models that are created using newspaper text and the speech transcriptions. The same procedure is followed for Flemish, but because of the lack of development data, no experiments are conducted to prove if mixing newspaper and transcription data is best for Flemish as well.

Language model	%WER	Significance (p)
Original LM	28.4	
Newspaper LM	28.6	0.549 (compared to original LM)
Newspaper/transcriptions LM	28.1	0.039 (compared to newspaper LM)

Table 7.1: *Results of experiments on the Dutch BN development set with the original LM and the two new LMs that are created using the N-Best data.*

The acoustic models

The data that are available for training the acoustic models are divided into four sets: a set for BN and CTS, both for Dutch and Flemish. The experiments described in the previous chapter were conducted with acoustic models trained on Dutch BN. The data are divided into the four sets because it is assumed that the four tasks are so different that mixing the data for acoustic model training would not result in better models. In general though, for statistical methods the rule of thumb is: 'more data is better'. Therefore, the question is raised if the Dutch and Flemish dialects are indeed so different that it is not beneficial to mix data for AM training. An experiment is conducted to find out if the performance of the system increases when all BN data (Flemish and Dutch) is used to train the model for the Dutch-BN task. The result of this experiment is shown in table 7.2. Apparently the Dutch and Flemish pronunciations are too different and in this case it does not help to simply add the Flemish data to the Dutch training set.

Acoustic model	%WER	Significance (p)
AM based on Dutch data	28.1	
AM based on all data	28.4	0.312 (compared to Dutch AM)
Mix of the first two models	27.8	0.046 (compared to Dutch AM)

Table 7.2: *Results of experiments on the Dutch BN development set with a model trained on Dutch data, a model trained on Dutch and Flemish data and a mix of these two models.*

During the training of the AM models, it was noticed that the likelihood on the training set increased for some phones and decreased for other phones when the Flemish data was added to the training set. It can be the case that for some phones it actually *does* help to add the Flemish data and for other phones it is better not to do this. In order to test this hypothesis, the overall likelihood on the Dutch-BN training data set is determined for each phone of both the AM trained on Dutch-BN and the

AM trained on all BN data. A third AM is then created by selecting the model with the highest likelihood for each phone. Note that although the first AM is trained solely on the Dutch-BN data and its likelihood is therefore tuned towards this set, there is quite a number of phones from the AM based on all data that score higher on the Dutch-BN data. The WER on the Dutch BN development set using this mixed AM is shown in table 7.2. This AM outperforms the AM trained on Dutch-BN with a significance of $p = 0.046$ and therefore the mix AM is used in the SHoUT submission for N-Best. For the three other tasks, the same procedure is followed for training the AM[2]. Table 7.3 contains statistics of each of the four AM files.

Task	Max #Gaussians	#Contexts	Total #Gaussians
BN-Dutch	60	1561	82687
BN-Flanders	60	1318	79068
CTS-Dutch	60	1251	74972
CTS-Flanders	60	1066	63870

Table 7.3: *Statistics of the final four sets of acoustic models used for the N-Best evaluation.*

The dictionaries

The pronunciations for all dictionaries used in this thesis are determined in two steps. First, a background dictionary with manually checked pronunciations is consulted and for the remaining words a graph-to-phoneme system is used to generate the pronunciations. This procedure is described in-depth in [Ord03].

The same procedure is used for the N-Best dictionaries. After generating the dictionaries a number of manual checks are performed. For the Dutch dictionary (used for the Dutch BN and CTS tasks), in total 2500 words are adjusted during this inspection. The increase in system performance after this manual work is limited but significant, as can be seen in table 7.4.

Dictionary	%WER	Significance (p)
Automatically generated	27.8	
Manually checked	27.5	0.004

Table 7.4: *Results of experiments on the Dutch BN development set with the automatically generated dictionary and the manually checked dictionary.*

Post-processing

The scoring method of N-Best is different in four aspects from the method used up until now in this work. First, scoring is performed case-sensitive. Also, strict rules are defined for the way numbers should be compounded. In the earlier experiments,

[2]Unfortunately, just before the submission deadline a bug was found in this procedure for CTS-VL and therefore for this task the acoustic model trained solely on Flemish CTS data is used.

the recognition of a number was considered correct even when it was not compounded correctly (for example: ‘honderd twee’, instead of: ‘honderdtwee’). Third, a lot of compounded words exist in the Dutch language. Words such as: ‘waterpolobal’, are written as multiple words in English (‘water polo ball’). In the earlier experiments the recognition of: ‘waterpolo bal’, was re-written to the correct word, but for N-Best these words will be considered incorrect. Finally, for the N-Best evaluation it is allowed to label filled pauses such as ‘eh’ or ‘uhm’, as non-lexical so that these words will be discarded during scoring. In order to face these four rules, a post-processing component is added to the ASR subsystem that handles the case of each word, the re-writing of numbers, compound restoration and the labeling of filled pauses.

The three words from the dictionary: ‘eh’, ‘uhm’ and ’mmh’, are marked as filled pause. Whenever these words are recognized, it will be labeled as such. For compound restoration, a list of possible compounds, abstracted from the Dutch newspaper corpus, is used. Whenever two words are recognized of which the compound is present in the list, the words are merged. The vocabularies are all case insensitive. For case-restoration, the disambigue tool from the sri-lm toolkit is used [Sto02]. Case sensitive language models are created for Flemish and Dutch, using the newspaper material. These models are used to map the lower-case recognitions to the correct case with the disambigue tool. The Dutch and Flemish dictionaries contain all possible numbers, but not in concatenated form. It contains the numbers ‘honderd’ and ‘drie’ but not ‘honderddrie’. Therefore a script is written that concatenates all numbers according to the rules of the evaluation. In section 7.1.3, each post-processing step is evaluated.

7.1.2 Evaluation results

The evaluation results are listed in table 7.5. Two aspects of the results stand out. The results of the broadcast news tasks are lower compared to the development set (For Dutch, 27.5 compared to 39.4% WER) and adaptation decreases the results for the CTS tasks. In the remainder of this section, in the post-evaluation analysis, these two aspects will be investigated.

Task	1^{st} iteration	2^{nd} iteration	Significance
	%WER	%WER	p
BN-Dutch	41.3	39.4	< 0.001
BN-Flanders	34.6	33.6	< 0.001
CTS-Dutch	60.4	60.7	0.136
CTS-Flanders	72.1	72.8	0.019

Table 7.5: *The N-Best evaluation results for each task and the significance of the difference between the first and second decoding iteration.*

7.1.3 Post-evaluation analysis

The results of the Dutch and Flemish broadcast news tasks are disappointing compared to the results of the Dutch BN development set. In part, the word error rates are higher because the data contain a good amount of interviews and discussions. The development data consists mainly of prepared studio speech. In table 7.6 the evaluation results of the Dutch BN task are shown for the four conditions with which the evaluation data was labeled. The word error rate of the clean studio condition is 26.3%. This is comparable to the results obtained on the development set (27.5%).

Audio condition	Number of words	SUB %	DEL %	INS %	WER %
Broadcast (F0)	7177	15.1	7.6	3.6	26.3
Spontaneous (F1)	10126	22.3	15.8	2.1	40.2
Telephone (F2)	3775	17.5	43.1	1.9	62.5
Degraded (F4)	2953	24.9	11.1	3.0	39.0

Table 7.6: *The N-Best evaluation results for the Dutch BN task. The results are shown for the main four audio conditions that are present in the task. The word error rate (WER) is divided into substitution (SUB), deletion (DEL) and insertion (INS) errors.*

The difference in evaluation and development data is not the only problem. After studying the evaluation results, it became clear that a high amount of speech was discarded by the segmentation subsystem that falsely labeled this speech as audible non-speech. Inspection of these segments revealed that the subsystem filtered all speech out of the system that was recorded over a telephone line. The deletion percentage of 43.1%WER in the F2 condition in table 7.6 is a clear indication of this problem. The development set did not contain any telephone speech and therefore this flaw in the system was not noted before. Next, a short explanation of this problem in the segmentation subsystem will be given.

Telephone speech in broadcast news

In chapter 4 the segmentation subsystem was described. This subsystem first identifies speech segments using a bootstrapping SAD component. Using this initial segmentation, a speech model, silence model and audible non-speech model are trained. For training the audible non-speech model, audio fragments with high energy levels that are classified as non-speech are used. Because in some cases the audible non-speech model is actually trained on speech fragments, the speech and non-speech models are compared and if they are considered similar, the audible non-speech model is discarded. The method failed because of two assumptions that are not valid in this case.

First, the bootstrapping is performed by a model based speech/silence segmentation component. This means that segments are only classified as speech when they fit the speech model well enough. If the audio conditions differ too much from the conditions of the training data for the speech model, speech segments might fit the more general silence model better, causing the segments to be classified as non-speech. This is what happened for the telephone speech.

Second, it is assumed that if the audible non-speech is trained with speech data, using the BIC method it is possible to detect that the speech model and audible non-speech model both contain speech so that the error can be fixed. This is indeed possible as long as the conditions of the speech of both models are similar enough. In this case the telephone speech with which the audible non-speech model was trained, did not match the speech from the speech model and the error was not fixed at all.

To avoid this problem a narrow-band/broadband detection subsystem can be applied before segmentation is performed. It is also possible to adjust the segmentation subsystem so that it is more robust for this channel problem. In the future work section of the next chapter, ideas for improving the segmentation subsystem will be given.

Task	BN model %WER	CTS model %WER	Significance p
BN-Dutch	35.5	34.9	< 0.001
BN-Flanders	33.5	31.7	< 0.001

Table 7.7: *The N-Best evaluation results for the Dutch BN task, where segments originally labeled as audible non-speech, are processed by the ASR subsystem using the telephone acoustic models. The significance levels are measured compared to the original submission (table 7.5).*

For this specific evaluation, where it is guaranteed that the audio fragments do not contain any audible non-speech, it is possible to interpret the results of the segmentation subsystem slightly different. In this case, all audio that is used for training the audible non-speech model is known to contain high energy levels and mismatch the acoustical conditions of the training data. The high energy levels indicate that the fragments are not silence and therefore *must* be speech. The obvious condition that does not match the training data is speech over telephone lines and therefore the segmentation results can be interpreted as a silence/studio-speech/telephone-speech classification. Table 7.7 contains the results of experiments on the Dutch and Flemish BN task where the audible non-speech segments were interpreted as being telephone speech. These segments are passed to both the broadcast news ASR subsystem and the CTS ASR subsystem[3]. The experiments show that indeed the best results are obtained by applying the CTS acoustic models. The word error rate of the telephone condition (F2) for the Dutch BN task is 39.6% (8.5%WER deletions) when using the CTS models.

Post-processing

In section 7.1.1, the post-processing steps were described that are added to the system for the N-Best evaluation. Table 7.8 contains the word error rates after each post-processing step for the Dutch BN task (where the telephone speech is decoded with the CTS acoustic models). If no post-processing would have been performed, the WER would be 38.6%. Table 7.8 shows that each of the steps improves this result.

[3]For this experiment, only one decoding pass is used. The models are not adapted for a second iteration.

Although the contribution of some post-processing steps is marginal, all improvements are significant with $p < 0.001$.

Post-processing step	%WER
No post-processing, scored case-sensitive	38.6
Filled pauses	37.9
Filled pauses and compounds	37.6
Filled pauses, compounds and case	35.3
Filled pauses, compounds, case and numbers	34.9
All post-processing, scored case-insensitive	33.8

Table 7.8: *Results of post-processing experiments on the N-Best Dutch-BN evaluation data. All experiments are significant with* $p < 0.001$.

If the task is scored case-insensitive, the WER is reduced with almost one percent. Although the case of the reference transcription is not correct in all places, this means that the case normalization step can be improved further.

7.1.4 Conclusions and discussion

The results of the N-Best evaluation show that on clean broadcast news speech, the system performance is comparable to the results during development. The WER on the spontaneous speech and the speech under degraded conditions is considerably higher. This confirms the conclusion in the previous chapter that the ASR subsystem is not yet robust enough against unknown audio conditions.

The evaluation revealed a weak spot of the segmentation subsystem. Although the behavior of the subsystem when segmenting BN audio that contains telephone speech is logical, it was not expected. It is possible to solve the problem by adding a broadband/narrowband classification subsystem, but it is preferred to adjust the segmentation subsystem so that it is possible to detect speech with various conditions that are all represented by bootstrap speech models. With the current subsystem it is only possible to detect the broadband speech represented by the bootstrap BN model, but it should be possible to add other models such as the CTS model during the bootstrapping step.

Compared to the SHoUT vocabularies, some participants of the N-Best evaluation used vocabularies with a considerably higher number of words. A vocabulary with 500K words was even used that has an out-of-vocabulary rate on the development set of only 0.5%. It would be interesting to test if the SHoUT performance can be improved by increasing the number of words in the vocabularies.

7.2 Surprise data: TRECVID

TRECVID is an annual benchmark for information retrieval of video collections. For 2007, the TRECVID collection consisted of video from a real-life archive of news-related genres such as news magazine, educational, and cultural programming. As in previous years, ASR transcripts of the data were provided as an optional information source for indexing. Apart from some English BN rushes (raw footage), the 2007 TRECVID collection, referred to as the TRECVID$_{07}$ collection, consisted of 400 hours of Dutch news magazine, science news, news reports, documentaries, educational programs and archival video. The files were provided by the Netherlands Institute for Sound and Vision[4].

As can be expected for a diverse content set such as the TRECVID$_{07}$ data, the audio and speech conditions vary enormously, ranging from read speech in a studio environment to spontaneous speech under degraded acoustic conditions. Furthermore, a large variety of topics are addresses and the material dates from a broad time period. Historical items as well as contemporary video fall within the range. (The former with poorly preserved audio; latter with varying audio characteristics, some even without 'intended' sound, just noise). The SHoUT system is evaluated on a set of TRECVID$_{07}$ recordings.

The N-Best evaluation showed that the SHoUT ASR subsystem is not especially robust for handling speech recorded in degraded conditions. Therefore, it is not expected that the accuracy on this data set will be high. The goal of this evaluation is to show that the SHoUT system is able to handle audio recordings that contain various types of audible non-speech fragments. It was shown in chapter 4 that the segmentation subsystem is able to reduce the SAD error for these kind of recordings, but it is not yet proven that the segmentation is suitable for use by the ASR subsystem. It is possible to create segmentations with short speech segments that have reasonably high SAD and diarization scores, but that are cutting-up words, making it impossible to recognize them correctly. The evaluation will show if the segmentations are suitable for decoding.

7.2.1 System description

For the evaluation on the TRECVID$_{07}$ data set, no complementary metadata is used and therefore it is not possible to apply supervised adaptation of the acoustic models for this task. The language model used at the first decoding pass is a trigram model similar to the NBest BN-Dutch model, except that it is trained with additional BN related texts (mainly auto-cues) that were not allowed to be used at the N-Best evaluation. For acoustic modeling the N-Best BN-Dutch models are used. In [HOdJ07] the language model as well as the employment of so called video-specific language models are discussed in-depth. The video-specific language models mentioned in this paper are not used for the experiments discussed in this section.

7.2.2 Evaluation results

For evaluation, a set of 12 fragments of 5 minutes each were randomly selected from the TRECVID$_{07}$ data and annotated manually. In chapter 4, the speech/non-speech results or this evaluation were already discussed. It was shown that the segmentation subsystem is able to remove various types of non-speech audio from the input stream.

[4]Netherlands Institute for Sound and Vision: `http://www.beeldengeluid.nl/`

In table 7.9 the segmentation and ASR results of the TRECVID$_{07}$ ASR evaluation are listed.

Segmentation and clustering	%SAD error	%WER 1^{st}	%WER 2^{nd}	%SUB 2^{nd}	%DEL 2^{nd}	%INS 2^{nd}
Manual	n.a.	63.3	62.3	39.0	19.9	3.4
Automatic	11.4	69.9	69.1	42.4	21.5	5.2

Table 7.9: *The results of the two decoding iterations of the SHoUT system applied on the TRECVID$_{07}$ ASR evaluation data. For the second iteration, the substitution (SUB), deletion (DEL) and insertion (INS) errors are specified.*

As can be seen in table 7.9, the word error rate of the SHoUT system on this evaluation set is high even if all non-speech is manually filtered out of the audio. On the other hand, unsupervised adaptation improves the result even at this WER level. Obviously the system performance decreases when the non-speech is not manually filtered out of the audio, but considering the high amount of audible non-speech, the degradation is limited. The absolute increase of the WER is 6.6%. Table 7.9 shows that this increase is not only caused by missed speech (deletions), but also because of an increased insertion rate. The SAD error of 11.4% consists of 8.3% missed speech and 3.2% false alarms (see chapter 4). These false alarms are responsible for the increase in insertion errors during decoding. Note that these percentages would be higher if the original BN segmentation system would have been used. The SAD error of this segmentation component is 20.3% of which 15.8% is due to missed speech and 4.5% is due to false alarms.

7.3 Conclusions

In this section it was investigated what the performance of the SHoUT system is on a task for which enough data is available to train models on, and what the performance is on a data set for which no training data is available. For this, two evaluations have been conducted. The first evaluation, the N-Best benchmark, consisted of four tasks: broadcast news and telephone speech for both Dutch and Flemish. For the analysis of this evaluation the focus was put on the Dutch broadcast news task. The second evaluation was the TRECVID task where the system was evaluated on audio of various sources.

The overall word error rate of the system on the Dutch BN task in the N-Best benchmark was higher than the error rate on the development set, but the performance on the part of the audio with studio BN conditions (F0) was similar to the performance on the development set. This result shows that the models of the system are properly trained and not over-fitted on the development data.

The BN tasks of the N-Best benchmark revealed a flaw in the segmentation subsystem. The subsystem is not able to classify telephone speech as speech in the absence of audible non-speech. The problem can be solved by adding a broadband/narrowband subsystem or by adjusting the initial step of the segmentation subsystem so that the bootstrapping segmentation does not only classify speech and silence represented by the main speech and silence models, but also takes into account speech represented by other speech models such as a CTS speech model.

The N-Best benchmark also showed that the system does not perform very well on audio with degraded conditions. This was expected as only a few measures were taken to make the decoder robust for mismatching training/evaluation conditions (CMN, VTLN and adaptation).

The TRECVID evaluation showed that the output of the segmentation and diarization subsystems is usable for the ASR subsystem even when large amounts of audible non-speech is present in the recording. The SAD error and diarization error percentages are both based on the percentage of time that an error is made. It is possible to create segmentations with short speech segments that have reasonably high SAD and diarization scores, but that cut-up words, making it impossible to recognize them correctly. The TRECVID evaluation showed that the loss in precision when performing ASR on the automatically obtained segments is limited.

CHAPTER 8

CONCLUSIONS

In the first chapter of this thesis a number of research goals and development requirements were formulated related to the design and implementation of the large vocabulary continuous speech recognition toolkit SHoUT. The SHoUT system was developed to require as few parameters and models tuned on training data as possible, so that the system is insensitive to any potential mismatch between training data and target audio. In the previous chapters the research questions were answered and the SHoUT system was evaluated. In this chapter, first the answers to the research questions will be summarized. Next, the development achievements will be evaluated. In the final section of this chapter recommendations for future research directions will be given.

8.1 Research goals and future directions

The SHoUT system described in this thesis consists of three subsystems: segmentation, diarization and ASR. In this section, for each of these subsystems the research questions that were defined in chapter 1 and the answers to these questions will be summarized. Also recommendations for future research, limited to the three subsystems, will be given. More general research directions will be discussed in section 8.3.

8.1.1 Segmentation

Many of the audio recordings that determined the focus of this research, do not only contain speech but also various kinds of non-speech sound. These audible non-speech fragments such as background noise, music or sound effects need to be separated from the speech segments. This observation led to the research question: *'How can all audible non-speech be filtered out of a recording without having any prior information about the type of non-speech that will be encountered?'*. In chapter 4 the segmentation subsystem was introduced that addresses this question. It is able to filter out all non-speech without the need of training statistical models on the various types of audible non-speech. For creating a bootstrapping segmentation, the subsystem only needs a statistical model for speech and for silence. In the various experiments described in chapter 4 it was shown that the segmentation subsystem is indeed able to properly segment out-of-domain speech (speech in another language than that of the bootstrapping model, section 4.7.2), filter music out of the recording (the IDIAP music evaluation, section 4.7.3) and successfully segment audio full of sound effects and background music (the TRECVID$_{07}$ collection, section 4.7.4).

Except for a small number of design parameters, the SHoUT segmentation subsystem does not involve any parameters that need tuning on a training set. This makes the algorithm robust for changing audio conditions. It was shown that it is not needed to use a high performing bootstrap segmentation component in order to obtain good final results, and therefore it is not a problem that the speech/silence models used in the bootstrap component are sometimes trained on data that mismatch the evaluation data. But because of these models, the second research problem: *'How can the system perform speech/non-speech segmentation without the use of statistical models created using training data?'*, is only solved in part. Yes, it is possible to create a segmentation system that does not deploy statistical models for audible non-speech, but future work is needed to prove that the bootstrapping component can easily be replaced by another component that does also not need to deploy models for speech and silence. A possible component that could be used instead of the bootstrapping component that requires speech and silence models, is a component that first finds all voiced fragments in the audio and then uses these fragments to train the initial speech model. Performing a number of training iterations while removing small gaps in the segmentation might be enough to also incorporate unvoiced speech into the model. This method is a promising approach especially because determining voiced speech regions can be done without the use of models and the majority of the audible non-speech of the resulting segmentation will actually be labeled as non-speech, but more research is needed to determine if this initial speech model is good enough to function properly in the successive steps of the segmentation subsystem.

In the evaluation chapter, chapter 7, the N-Best evaluation revealed that speech recorded over a telephone line is not classified as speech by the segmentation subsystem when the recording does not contain any audible non-speech. This is because of two reasons. First, because of the absence of any real audible non-speech, the system will select telephone speech segments to train the 'sound' model. Second, because the characteristics of the telephone channel and the broadband studio channel are too different, in the final step of comparing the sound model and the speech model, the sound model is not discarded. This problem can be solved by generalizing the segmentation subsystem. The current approach is to segment the recording into two known classes (speech and silence) and one unknown class (audible non-speech). The models for the known classes are re-trained so that they fit the recording the best. For the new method, the number of known classes is increased. For each class, a bootstrap model is needed that is used in the initial segmentation step. Then, similar to the current approach, all audio that does not fit the known classes very well is used to train the 'sound' model. In the remaining steps, the models for the known classes are re-trained and a final Viterbi run is performed. To solve the telephone line problem, one of the new models will be the acoustic model for telephone speech.

A problem related to SAD that is not yet addressed by the proposed system is detecting and discarding foreign speech fragments. Feeding foreign speech into the ASR subsystem will obviously influence its performance negatively as the ASR subsystem can only recognize speech of the target language. Unfortunately, as was shown by the experiments on the RT06s conference meeting data, the SHoUT SAD subsystem will classify speech from foreign languages as speech. A solution to this problem is to apply a language detection subsystem directly after the SAD subsystem. Speech from a language for which an ASR system is available can be passed to that system, while speech of other languages can be discarded. Although this solution seems logical and straightforward, another solution is also interesting to investigate: The creation of language-specific speech models. For each language a SAD speech model is trained and each of these models is applied during the bootstrapping step of the generalized segmentation subsystem. Similar to splitting telephone speech and broadband speech it might be possible to categorize multiple languages.

8.1.2 Speaker diarization

The first research question concerning speaker diarization, defined in chapter 1, is: *'How can a speaker clustering system be created that does not require any statistical models created using training data?'*. This question was addressed during the development of the speaker diarization subsystem. The diarization subsystem does not need any training data for the creation of statistical models. Instead it randomly cuts up the recording and trains models on the recording that is being processed itself. By performing a number of Viterbi re-alignments while training the models, each speaker gradually captures his or hers own model.

The second question related to speaker diarization: *'How can the proposed speaker clustering system be adjusted so that it is able to process long recordings with reasonably computational effort?'*, is also answered in this thesis. Although it must be noted that the two solutions,SHoUT$_{DCM}$ and SHoUT$_{D07*}$, both speed-up the process but also decrease the performance. In order to understand why the SHoUT$_{DCM}$ system did not perform as well as the original diarization subsystem, future research is needed. A thorough analysis such as performed for SHoUT$_{D06}$ can reveal the aspects that need improvement. Due to time constraints it was not possible to perform an analysis for both systems. Instead, the SHoUT$_{D07*}$ system was developed that is closely related to the SHoUT$_{D07}$ system. With the use of a single parameter, it is possible to decrease the real-time factor of this subsystem with a slight decrease in performance. For short recordings though, the SHoUT$_{D07*}$ system is identical to the SHoUT$_{D07}$ system.

It is interesting to plan future research for diarization of long recordings at the SHoUT$_{DCM}$ approach, but also at two other approaches. First, the process could be made faster by combining the SHoUT$_{DCM}$ approach and the SHoUT$_{D07}$ approach. For long recordings, the first iterations could be performed by SHoUT$_{DCM}$ while the final iterations are performed by SHoUT$_{D07}$. The experiments from section 5.5.1 indicate that SHoUT$_{DCM}$ seems weak in deciding the optimal number of clusters, but it works fine in clustering the initial models.

The second possible approach to diarization of long recordings is related to the approach taken for SAD. Instead of processing the entire recording at once, it could be cut up in chunks and each chunk could be processed individually. Although for SAD it is relatively easy to re-combine the chunks, for diarization this step is not so straightforward. It is not that easy to determine which speaker model from one chunk matches the model of another chunk. If a method *is* found that can match the correct speaker models of the various chunks, it is possible to process recordings of infinite length. It would also make tracking of speakers over multiple recordings straightforward.

8.1.3 Automatic speech recognition

From a software engineering point of view, the ASR subsystem is the most complex of all three subsystems. This is the reason that in the chapter about ASR, chapter 6, a number of development issues has been presented. Especially the implementation of a modular system received special attention in the first section of chapter 6 and also the implementation and evaluation of a number of techniques for robust ASR was discussed and the question: *'Which methods can be applied to make the decoder insensitive for a potential mismatch between training data and target audio?'*, was addressed. The three methods: cepstrum mean normalization, vocal tract length normalization and structured maximum a posteriori linear regression, all proved to reduce the word error rate significantly.

In dealing with the development requirements, a very specific research question was encountered: *'How can full language model look-ahead be applied for decoders with static pronunciation prefix trees?'* Language Model Look-Ahead (LMLA) is a very helpful technique in managing the computer resources needed by the ASR system. Unfortunately, it is not straightforward how to use this technique with the system architecture that was chosen in order to fulfill the development requirements. In chapter 6 this problem was addressed and a method to efficiently use language model look-ahead in the SHoUT decoder or in any other decoder using a single pronunciation prefix tree was discussed and evaluated. It was shown that LMLA speeds up the decoder considerably without loss of recognition precision. It was also shown that in the SHoUT decoder, the full LMLA architecture outperforms unigram look-ahead.

The unigram LMLA system was slower than the full LMLA system (1.35 times) and as expected the computational cost of the system without *any* LMLA was the highest (2.4 times as slow as the optimal system). The fact that unigram LMLA already provides a considerable speed-up, and that it is less complex to implement than full LMLA, could be a consideration to chose for unigram LMLA. Also, note that the reported real-time factor results are closely related to the implementation of the SHoUT decoder and that it is possible that, if implemented in another decoder, the RTF gain of the full LMLA system compared to the unigram LMLA system is less distinct. Given this caveat, the experiments with the SHoUT decoder are very promising and it is believed that full LMLA using the proposed data architecture will also improve the real time factor of other token passing decoders.

Some decoders use pre-compiled caches for LM probability look-up of the most occurring words. This helps because these words are used considerably more often than the remaining words and therefore have a high probability of being looked up. In the SHoUT decoder, no cache is being used, but instead a very efficient LM look-up method is implemented that reduces a regular n-gram query to calculating the key for a minimum perfect hash table and using this key to directly access the probability. A cache might be useful for speeding up the calculation of the key, but the effect of this speed-up will be highly limited.

8.1.4 The sum of the three subsystems: the full SHoUT system

As described in chapter 7, the three subsystems were combined and the full system was tested on two benchmarks. The first benchmark, N-Best, consisted of four tasks: broadcast news and telephone speech for both Dutch and Flemish. For the analysis of this evaluation the focus was put on the Dutch broadcast news task. The second evaluation was the TRECVID task where the system was evaluated on audio of various sources.

The ASR performance of the system on the Dutch BN task in the N-Best benchmark was not as good as on the development set, but the performance on the part of the audio with studio BN conditions (F0) was similar to the performance on the development audio. This result shows that the models of the system are properly trained and not over-fitted on the development data. Also, it proves that for BN audio conditions, the three subsystems are able to work together properly. For degraded audio conditions, the benchmark showed that the system does not perform very well. This was expected as only a few measures were taken to make the decoder robust for mismatching training/evaluation conditions (CMN, VTLN and adaptation).

The TRECVID evaluation showed that the output of the segmentation and diarization subsystems is usable for the ASR subsystem even when high amounts of audible non-speech is present in the recording. The SAD error and diarization error percentages are both based on the percentage of time that an error is made. It is possible to create segmentations with short speech segments that have reasonably high SAD and diarization scores, but that cut-up words, making it impossible to recognize them correctly. The TRECVID evaluation showed that the loss in precision when performing ASR on the automatically obtained segments is limited.

8.2 Development goals

In chapter 1, a number of development goals were defined. The source code created for this research needs to be transparent so that it is easy to understand what each line of code does. The language specific information needs to be stored in binary files and not in source code. The software should be set-up modular so that a task can be performed stand-alone. It should be easy to replace functional parts of the software (such as algorithm steps) with alternative implementations. And finally, general purpose source code (for example source code for handling Gaussians) needs to be re-usable for each module.

These goals were kept in mind during development of the SHoUT toolkit and it is the believe of the author that they are all reached, but for non of the goals this is actually tested. A group of software engineers is needed to evaluate the quality of the software and determine if the source code is actual transparent, modular and re-usable. Instead of hiring a group of software engineers, the toolkit is simply made available on the internet under an open source license. Every software engineer can now determine for herself what the quality of the source code and the manual is.

Although it is not proven that each of the development goals are reached, the ease with which some of the algorithms that are described in this thesis could be implemented, indicates that the source code is indeed transparent, set up modular and easily re-usable. For example, both the segmentation as the diarization software have been built on top of the decoder software described in chapter 6. These applications make use of the HMM and the token passing modules while the language model is omitted and the structure of the lexical tree is modified. Also the software proved to be easily extended with the extra module needed for the ASR research topic: efficiently managing the decoder's search space. Finally, it was no problem to temporarily replace steps of the diarization subsystem with Oracle components during the analysis of this subsystem.

8.3 Extending the horizon for SHoUT

A number of ideas for future work on the three subsystems has been presented in section 8.1. The segmentation subsystem can be generalized so that it is possible to classify more known classes (such as telephone speech or language). For the diarization subsystem future research is needed to improve the SHoUT$_{DCM}$ system and to create a method that can link the speaker clusters from multiple recordings to each other, so that the process can be done in parallel or speakers can be tracked across multiple recordings. For the ASR subsystem, additional methods could be integrated to reduce the mismatch between training data and evaluation data. Also the ASR decoder needs to be optimized so that its speed is comparable to other state-of-the-art decoders.

These future work proposals all extend the research described in this thesis in a way so that the overall system is improved gradually for large vocabulary continuous speech recognition. These improvements are needed, but with the current state of the framework, it is also possible to apply SHoUT for other tasks than LVCSR. In this section research directions in the field of automatic speech recognition and spoken document retrieval are investigated.

8.3.1 Automatic speech recognition

The software framework described in this thesis is created as a research platform for SDR, but SHoUT can also be deployed for other research and application development. With a few adjustments, the SHoUT toolkit is suitable to be used in various Human Computer Interaction (HCI) research projects. Some very interesting examples of

HCI projects can be found close to home, at the Human Media Interaction (HMI) group at the University of Twente. For example in [Bui08], research on decision support systems for optimizing dialog management strategies is described. These dialog management systems use speech interfaces as one of their input sources[1]. In order to apply SHoUT to these kind of projects, some adjustments are needed. For example, the current version of the decoder is not able to handle finite state grammars, the standard for a lot of dialog systems, but it is not difficult to replace the language model component with an implementation for finite state grammars. By doing this, it is not only possible to assist research in the field of HCI, but also to investigate the interaction of ASR with dialog management systems. This development would be interesting for future work in complex dialog management systems.

None of the algorithms described in this thesis are developed to handle *online* audio streams. Online systems are able to perform a task on an audio stream directly, without the need of first recording an entire session. The building blocks for developing such online methods are available in the SHoUT toolkit and it would be interesting to investigate if the three subsystems can be adjusted so that they are able to process data online. This would, for example, make it possible to apply the subsystems for direct monitoring of meetings or for the online generation of subtitles. The building blocks for implementing such applications are available and with some engineering efforts, SHoUT can be used for various automatic speech processing projects outside the spoken document retrieval domain.

8.3.2 Spoken document retrieval

The purpose of the ASR system described in this work is to solve the representation mismatch between speech in audio recordings and written words of the query in SDR systems. In the context of SDR, some interesting research topics can be studied.

Retrieval issues

Although the word error rate is a good metric for measuring the performance of ASR systems, for use in SDR systems it should not be the only metric used. For example, out-of-vocabulary words increase the WER only marginally as long as the vocabulary is large enough, but from an SDR perspective, the system performance degrades significantly when out-of-vocabulary words need to be searchable.

Recent studies apply hybrid ASR systems to solve the out-of-vocabulary problem that LVCSR systems suffer from. Sub-word based recognizers are combined with LVCSR systems so that even words that are not in the vocabulary can be searched [MMRS08, ESS08, SFB08].

Focusing on the SDR performance instead of solely on the ASR performance also payed of in [YTS08]. The SDR performance was increased considerably by processing word lattices instead of the single best ASR output.

Performing this kind of research on SDR issues, improving SDR performance and addressing the out-of-vocabulary problem, is possible with some minor exten-

[1]Other input sources include facial expression and gesture information.

sions of the SHoUT toolkit and steps in this research direction are being taken, e.g. in [vdWH08].

Applications

The current framework is able to provide time aligned speech transcripts within an SDR application and it can also provide speaker label information and classify speech, silence and audible non-speech. With some extensions of the SHoUT toolkit it is possible to create more types of metadata and create new SDR applications.

As mentioned earlier, the diarization subsystem can be used for speaker tracking across multiple recordings if a good method can be found for linking the clusters of the recordings to each other. Once this is possible, various kinds of available information can be linked to each speaker cluster. In addition to the speech transcripts, it can be useful to automatically determine a speakers' gender or age(group), but with the use of ASR transcripts or external sources such as the internet, it might even be possible to extract the name of the speaker. For example, a movie star might never be called by his own name in the movie itself, but his name might be mentioned in a TV interview or at least in the TV guide that describes the interview. This kind of information linking is in part already possible. Within the MultimediaN project (see appendix B) for example, broadcast news topics were linked to relevant newspaper articles. With the extension of the speaker diarization subsystem, it will become possible to apply information linking also at speaker level.

In the current version of the SHoUT SAD subsystem, all audible non-speech is filtered out of the recordings. It is imaginable that these non-speech fragments are not thrown away, but clustered automatically in a similar fashion as is done in speaker diarization. If the right features are chosen, it will become possible to automatically cluster all kinds of sounds and if sound clusters of multiple recordings can be linked together, large sound byte databases can be created. After manually or perhaps automatically labeling the clusters, it becomes feasible to perform data mining and, for example, determine the type of a movie from its sound effects or calculate how often a meeting is disturbed by nearby construction work. It might even be possible to link the speaker tracking system to this sound classification system and determine in what kind of movies a specific movie star generally acts.

By linking together various multimedia sources with the use of automatically extracted metadata such as text transcripts, speaker information and sound classifications, numerous interesting new spoken document retrieval applications can be created. It is not unthinkable that this new technology would change the story in the first section of chapter 1 as follows.

Unforgettable memories

...I'm just finished explaining what a pointer is when my car navigation system tells me that I have reached my destination. The technology saved my day. And we're just getting started.

After dinner we tell the computer that we want to watch the one-hour version of our Holiday, uncensored and without extra tourist facts. The version that I showed my parents was fun, but tonight I don't want to watch a documentary about bridges and prisons. The system understands this perfectly and it creates a nice road movie for us. Even the title song is great. It's from the CD that we've been playing all summer. Eating our fruit cocktail we watch Chuck setting half of Dave's land on fire. Suddenly the movie is interrupted by a video message that Chuck and David left for us. The computer added some pictures from the web showing Dave's new house. After the movie we shoot our own message for their Holiday album and we watch some facts that the computer calculated for us. We skip the part about calories, hours of sun exposure and beer consumption, but the list of comparable Holiday destinations gets my full attention. Especially because the system cross-linked them with interesting conferences. Australia, here I come!

Appendices

APPENDIX A

DATA COLLECTIONS

A number of collections was used for training, development and evaluation of the SHoUT system and its subsystems. In the following sections, these collections will be described.

A.1 Spoken Dutch Corpus

The Spoken Dutch Corpus ('Corpus Gesproken Nederlands', CGN) is a collection of Dutch and Flemish speech recordings annotated mostly on a word basis and for part of the corpus, phone-based transcriptions are available. The corpus is divided into 15 components. Each component contains recordings of a specific type such as broadcast news shows, telephone recordings or ceremonious sermons [Oos00].

For this research, CGN was used for the training of acoustic models. The broadcast news models were trained with the recordings of interviews, live commentaries, news reports, broadcast news shows and broadcast commentaries (components f,i,j,k and l). For Dutch the total length of the recordings is 99.4 hour. For Flemish it is 52.9 hour. The telephone models were trained on telephone speech from the components c and d. For Dutch the total amount of data was 92.0 hour and for Flemish it was 64.0. Note that the recording time is inclusive silence regions.

A.2 N-Best

In 2006 the research project: 'Northern and Southern Dutch Benchmark Evaluation of Speech recognition Technology' (N-Best) was started. N-Best aims at setting up the infrastructure for a benchmark evaluation in large vocabulary speech recognition for the Dutch language, and at conducting the evaluation. This evaluation, the first one for Dutch ASR, focuses on two tasks: broadcast news and Conversational Telephone Speech (CTS). Within these tasks, both Northern and Southern Dutch as it is spoken in the Netherlands and Flanders (Belgium) respectively, are evaluated.

The N-Best evaluation data consists of 134 minutes of broadcast news recordings for Dutch and 129 minutes of broadcast news recordings for Flemish. The telephone speech task consists of 174 minutes of Dutch speech (counting both channels) and 153 minutes of Flemish speech. The N-Best benchmark was used to evaluate the full SHoUT system.

The development data for N-Best was mainly data from CGN, but the N-Best project also contributed a number of Dutch broadcast news recordings (in total 30 minutes long). These recordings were used as part of the development set for the ASR subsystem.

A.3 Twente news corpus

The Twente News Corpus (TwNC) is a corpus of Dutch newspapers that was used for training language models. The corpus also contains a number of broadcast news recordings transcribed on a word basis. These recordings, together with the broadcast news recordings annotated by the N-Best project, were used as development set for the ASR subsystem. In total, the development set consists of 84 minutes of speech from in total 114 speakers.

For the development of the speech activity detection subsystem, one of the recordings of this test set was forced aligned using the ASR subsystem, so that the segments containing speech were known precisely. Manually annotated speech/non-speech transcriptions often contain silence at the edges of each speech segment. These small regions of silence are falsely classified as speech and can influence the SAD error rate considerably. This recording, the recording of 27/09/2006 was used to evaluate the SAD subsystem and it is 24 minutes long.

A.4 TRECVID$_{07}$ data set

TRECVID is an annual benchmark for information retrieval of video collections. For 2007, the TRECVID collection consisted of video from a real-life archive of news-related genres such as news magazine, educational, and cultural programming provided by the Netherlands Institute for Sound and Vision. As in previous years, ASR transcripts of the data were provided as an optional information source for indexing. Apart from some English BN rushes (raw footage), the 2007 TRECVID collection, referred to as TRECVID$_{07}$consisted of 400 hours of Dutch news magazine, science news, news reports, documentaries, educational programs and archival video. The files were provided by the Netherlands Institute for Sound and Vision[1].

As can be expected for a diverse content set such as the TRECVID$_{07}$ collection, the audio and speech conditions vary enormously, ranging from read speech in a studio environment to spontaneous speech under degraded acoustic conditions. Furthermore, a large variety of topics are addresses and the material dates from a broad time period. Historical items as well as contemporary video fall within the range. (The former with

[1]Netherlands Institute for Sound and Vision: `http://www.beeldengeluid.nl/`

poorly preserved audio; latter with varying audio characteristics, some even without 'intended' sound, just noise).

From the 400 hours of recordings, 12 fragments of 5 minutes each were randomly selected and annotated manually. This set of fragments, referred to as the TRECVID$_{07}$ ASR evaluation data set was used to evaluate SAD subsystem in chapter 4 and the full SHoUT system in chapter 7.

A.5 Rich Transcription benchmark for meetings

The National Institute for Standards and Technology (NIST) annually organizes a benchmark for rich transcription of recordings in the meeting domain. Each year, participants of the benchmark contribute meeting recordings to form the conference room meeting test set. Each conference room meeting is recorded with multiple far-field microphones and with a close-talking microphone for each person in the meeting. All meetings are in English. Note that next to the conference room test set, there is also a test set of lectures, but that for this research only the conference room recordings are used.

For speaker diarization, two tasks are defined: the Multiple Distant Microphone (MDM) task and the Single Distant Microphone (SDM) task. For the MDM task, all available far-field microphone recordings are allowed to be used, while for the SDM task, only one far-field microphone that is picked by NIST is allowed to be used.

For this research, the conference room test sets of 2005, the Rich Transcription 2005 Spring (RT05s) test set, was used for development of the SHoUT$_{D06}$ and SHoUT$_{DCM}$ speaker diarization subsystems. These subsystems were evaluated on the 2006 data set (RT06s). The SHoUT$_{D07}$ was evaluated on the set from 2007 (RT07s) and developed on 21 meetings from various years (see table 5.7 from chapter 5).

A.6 The IDIAP speech/music evaluation set

The IDIAP speech/music evaluation set is a set of four audio recordings that contain English broadcast news shows interleaved with various genres of music [AMB03]. The first file contains speech and music fragments of fifteen seconds each. The second file contains fragments of varying lengths but overall with the same amount of speech as music. The third file contains more speech than music while the fourth file contains more music.

For this research, the IDIAP speech/music evaluation set is used for the evaluation of the speech activity detection subsystem.

APPENDIX B

PROJECTS AND DEMONSTRATORS

The work reported in this thesis was supported and deployed by a number of research projects. An overview of these projects and descriptions of demonstrators developed for these projects will be given in the following sections.

B.1 MultimediaN

In the MultimediaN project, Dutch scientific groups work together with industrial and other non-profit institutions on multimedia research topics. A broad spectrum of topics is addressed from low level feature extraction in pictures, video or audio to complex search technology. The research in MultimediaN is organized in a number of projects and the work reported in this thesis is part of the N5 project: Semantic Multimedia Access. MultimediaN is supported by the Dutch government under the BSIK program.

Together with other N5 team members, a demonstrator has been created in which the SHoUT toolkit is used. The demo, called StreetTivo, consists of a network of personal hard disk video recorders with which it is possible to index multimedia collections in a peer to peer manner. By combining the computational power of the machines of all StreetTivo users, it is possible to quickly perform automatic speech recognition of various television shows.

In cooperation with the Willem Frederik Hermans Institute (WFHi) and the Digitale Bibliotheek voor de Nederlandse Letteren (DBNL), another application has been created for search in radio and television interviews with the famous Dutch novelist Willem Frederik Hermans. This search engine is accessible online through the web portal: `/www.willemfrederikhermans.nl`.

More information about MultimediaN and the StreetTivo demonstrator can be found on the MultimediaN site: `www.multimedian.nl`.

B.2 AMI and AMIDA

In the Augmented Multi-party Interaction (AMI) project and in its successor the Augmented Multi-party Interaction with Distant Access (AMIDA) project, applied research is performed for increasing the productivity of meetings. In AMI research was focused on the development of meeting browsers, tools that allow users to search information in recorded meetings. In AMIDA, this work is continued and the research is extended to two new areas. The first new research direction is content linking, automatically finding information that is relevant for an ongoing meeting. The second direction is developing new technology for people who are using a telephone or other technology to connect to a meeting that otherwise they could not attend. The AMI and AMIDA projects are funded by a grant (project number IST-2002-506811) from the European Community Framework 6 Programme in Information Society Technology (IST).

The work on speaker diarization reported on in this thesis was in part supported by the AMI and AMIDA projects. Thanks to the training program of AMIDA I have been able to visit the International Computer Science Institute (ICSI), in Berkeley for half a year and to attend the NIST benchmark for Rich Transcription of meetings in 2006 and in 2007 (RT06s and RT07s).

For more information on the AMI project: `/www.amiproject.org`.

B.3 CHoral

The CHoral project aims at the development of spoken document retrieval technology for the disclosure of oral history collections. The project focuses on the development of a methodological framework for handling and use of multimedia oral history content for historical research. The CHoral project is supported by the Dutch national research program for Continuous Access to Cultural Heritage (CATCH).

In cooperation with the Netherlands Institute for War Documentation (NIOD) a demonstrator has been created for search in interviews with survivors of the second world war concentration camp Buchenwald. The SHoUT system was used to generate speech transcriptions of these interviews. The demonstrator is accessible at: `www.buchenwald.nl` (in Dutch).

B.4 N-Best

The Northern and Southern Dutch Benchmark Evaluation of Speech recognition Technology (N-Best) project aims at setting up the infrastructure for a benchmark evaluation in large vocabulary speech recognition for the Dutch language, and at conducting such an evaluation. In 2006 the Dutch research programme STEVIN granted funding to the project and in 2008 the first benchmark for Dutch large vocabulary continuous speech recognition has been held. The SHoUT system was used for the primary submission of the team from the University of Twente. In chapter 7 this submitted system has been discussed.

B.5 TRECVID

The TREC Video Retrieval Evaluation (TRECVID) is an annual video retrieval benchmark sponsored by the National Institute of Standards and Technology (NIST) with additional support from other U.S. government agencies. The goal of this conference series is to encourage research in information retrieval by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results. For the benchmarks of 2007 and 2008, the Netherlands Institute for Sound and Vision has provided 400 hours of video from a real-life archive of news-related genres such as news magazine, educational, and cultural programming (see appendix A). The SHoUT system was used to generate speech transcriptions of these videos.

B.6 MESH

Multimedia Semantic Syndication for Enhanced News Services (MESH) will apply multimedia analysis and reasoning tools, network agents and content management techniques to extract, compare and combine meaning from multiple multimedia sources and produce advanced personalized multimedia summaries, deeply linked among them and to the original sources to provide end users with an easy-to-use multimedia mesh concept, with enhanced navigation aids. A step further will empower users with the means to reuse available content by offering media enrichment and semantic mixing of both personal and network content, as well as automatic creation from semantic descriptions. Encompassing all the system, dynamic usage management will be included to facilitate agreement between content chain players (content providers, service providers and users). In a sentence, the project will create multimedia content brokers acting on behalf of users to acquire, process, create and present multimedia information personalized (to user) and adapted (to usage environment). These functions will be fully exhibited in the application area of news, by creation of a platform that will unify news organizations through the online retrieval, editing, authoring and publishing of news items.

The MESH project uses the SHoUT system for segmentation, diarization and automatic speech recognition of the broadcast news recordings.

B.7 MediaCampaign

Knowledge about which competitor company has invested how much money in a specific media campaign is very important for the highest management level of companies. Such information is gathered through global advertisement expenditure measurement, which is performed by media monitoring companies. This type of business intelligence is a very complex task, which is currently performed manually and therefore is very expensive.

In the MediaCampaign project, research is performed on automatically gathering knowledge about media campaigns. MediaCampaign's scope is on discovering, interrelating and navigating cross-media campaign knowledge and to automate a large degree of the detection and tracking of media campaigns on television, Internet and in the press. For the pilot system developed within the project, the focus is on a concrete example for a media campaign: advertisement campaigns. For this pilot, the SHoUT toolkit is deployed to perform automatic speech recognition.

BIBLIOGRAPHY

[ABD$^+$02] A. Adami, L. Burget, S. Dupont, H. Garudadri, F. Grezl, H. Hermansky, P. Jain, S. Kajarekar, N. Morgan, and S. Sivadas. Qualcomm-icsi-ogi features for asr. In *proceedings of ICSLP*, 2002.

[ABLM02] Jitendra Ajmera, H. Bourlard, I. Lapidot, and I. McCowan. Unknown-multiple speaker clustering using HMM. In *proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Denver, Colorado, USA, 2002.

[AMB03] Jitendra Ajmera, Iain McCowan, and Hervé; Bourlard. Speech/music segmentation using entropy and dynamism features in a HMM classification framework. *Speech Communication*, 40(3):351–363, 2003.

[AMSM96] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul. A compact model for speaker-adaptive training. In *proceedings of the Fourth International Conference on Spoken Language, 1996. ICSLP'96*, volume 2, pages 1137–1140, 1996.

[Ang06] Xavier Anguera. *Robust Speaker Diarization for Meetings.* PhD thesis, Universitat Politecnica De Catalunya, 2006.

[AW03] Jitendra Ajmera and Chuck Wooters. A robust speaker clustering algorithm. US Virgin Islands, USA, December 2003.

[AWP07] Xavier Anguera, Chuck Wooters, and J. Pardo. Robust speaker diarization for meetings: ICSI RT06s evaluation system. In *Machine Learning for Multimodal Interaction (MLMI)*, volume 4299 of *Lecture Notes in Computer Science*, Berlin, October 2007. Springer Verlag.

[AWPA06] Xavier Anguera, Chuck Wooters, Barbara Peskin, and Mateu Aguiló. Robust speaker segmentation for meetings: The ICSI-SRI spring 2005 diarization system. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, pages 402–414, Berlin, February 2006. Springer Verlag.

[BAHU$^+$99] P. Beyerlein, X. Aubert, R. Haeb-Umbach, M. Harris, Dietrich Klakow, A. Wendemuth, Sirko Molau, Michael Pitz, and A. Sixtus. The Philips/RWTH system for transcription of broadcast news. In *Proceedings of the 1999 DARPA Broadcast News Workshop*, 1999.

[BCK03] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice, Second Edition.* Addison Wesley, 2003.

[Bin99] Robert V. Binder. *Testing object-oriented systems: models, patterns, and tools.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Bui08] Trung H. Bui. *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes.* PhD thesis, University of Twente, 2008.

[Cas04] Steve Cassidy. The macquarie speaker diarisation system for RT04S. proceedings of the NIST RT04s Evaluation Workshop, Montreal, Canada, May 2004.

[CCE$^+$99] Gary Cook, James Christie, Dan Ellis, Eric Fosler-Lussier, Yoshi Gotoh, Brian Kingsbury, Nelson Morgan, Steve Renals, Tony Robinson, and Gethin Williams. An overview of the SPRACH system for the transcription of broadcast news. In *Proceedings of the 1999 DARPA Broadcast News Workshop*, 1999.

[CDGM02] A. Cardenal, J. Dieguez, and C. Garcia-Mateo. Fast lm look-ahead for large vocabulary continuous speech recognition using perfect hashing. In *proceedings ICASSP 2002*, pages 705–708, Orlando, USA, 2002.

[CEG$^+$99] S. S. Chen, E. M. Eide, M. J. F. Gales, R. A. Gopinath, D. Kanevsky, and P. Olsen. Recent improvements to IBM's speech recognition system for automatic transcription of broadcast news. In *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*, Washington, DC, USA, 1999.

[CG98] Shaobing S. Chen and P. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, Virginia, USA, 1998.

[CGD$^+$07] Paul Clough, Michael Grubinger, Thomas Deselaers, Allan Hanbury, and Henning Mller. Overview of the imageclef 2006 photographic retrieval and object annotation tasks. In *Evaluation of Multilingual and Multi-modal Information Retrieval – Seventh Workshop of the Cross-Language Evaluation Forum, CLEF 2006*, LNCS, Alicante, Spain, September 2007.

[Cha97] Lin Chase. Blame assignment for errors made by large vocabulary speech recognizers. In *proceedings Eurospeech '97*, pages 1563–1566, Rhodes, Greece, 1997.

[CHE+06] Murray Campbell, Alexander Haubold, Shahram Ebadollahi, Milind R. Naphade, Apostol Natsev, Joachim Seidl, John R. Smith, Jelena Tei, and Lexing Xie. Ibm research trecvid-2006 video retrieval system. In *NIST TRECVID Workshop*, Gaithersburg, MD, November 2006.

[CHJ+06] Shih-Fu Chang, Winston Hsu, Wei Jiang, Lyndon Kennedy, Dong Xu, Akira Yanagawa, and Eric Zavesky. Columbia University TRECVID-2006 Video Search and High-Level Feature Extraction. In *NIST TRECVID Workshop*, Gaithersburg, MD, November 2006.

[CHM92] Zbigniew J. Czech, George Havas, and Bohdan S. Majewski. An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5):257–264, 1992.

[Chu03] K. W. Church. Speech and language processing: Where have we been and where are we going? In *proceedings of Interspeech*, Genève, Switzerland, September 2003.

[CKA95] Jordan Cohen, Terri Kamm, and Andreas G. Andreou. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. *Journal of the Acoustical Society of America*, 97(5):3246–3247, 1995.

[CNZ+06] T.-S. Chua, S.-Y. Neo, Y. Zheng, H.-K. Goh, Y. Xiao, M. Zhao, S. Tang, S. Gao, X. Zhu, L. Chaisorn, and Q. Sun. Trecvid 2006 by nus-i2r. In *NIST TRECVID Workshop*, Gaithersburg, MD, November 2006.

[DDCW00] Kris Demuynck, Jacques Duchateau, Dirk Van Compernolle, and Patrick Wambacq. An efficient search space representation for large vocabulary continuous speech recognition. *Speech Communications*, 30(1):37–53, 2000.

[DJLW06] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. Technical report, The Pennsylvania State University, 2006.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[DM80] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech, and Signal Processing, IEEE Transactions on*, 28(4):357–366, Aug 1980.

[ESS08] Stefan Eickeler, Jochen Schon, and Daniel Schneider. Towards large scale vocabulary independent spoken term detection: Advances in the fraunhofer iais audiomining system. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[FA07] J. Fiscus and J. Ajot. The Rich Transcription 2007 Speech-To-Text (STT) and Speaker Attributed STT (SASTT) Results. In *Presentation at NIST's Rich Transcription 2007 Meeting Recognition Workshop*, 2007.

[FAG08] Jonathan G. Fiscus, Jerome Ajot, and John S. Garofolo. The rich transcription 2007 meeting recognition evaluation. In *Multimodal Technologies for Perception of Humans*, Lecture Notes in Computer Science, Berlin, 2008. Springer Verlag.

[FFKW99] M. Finke, J. Fritsch, D. Koll, and A. Waibel. Modeling and efficient decoding of large vocabulary conversational speech. In *proceedings of Eurospeech'99*, pages 467–470, Budapest, Hungary, 1999.

[Fis97] Jonathan G. Fiscus. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (rover). In *proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, Santa Barbara, CA, 1997.

[FS07] Corinne Fredouille and Grégory Senay. Technical improvements of the E-HMM based speaker diarization system for meeting records. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, Berlin, January 2007. Springer Verlag.

[GAV00] J. Garofolo, G. Auzanne, and E. Voorhees. The trec spoken document retrieval track: A success story. In *proceedings of the Recherche d'Informations Assiste par Ordinateur: ContentBased Multimedia Information Access Conference*, 2000.

[GC89] L. Gillick and S.J. Cox. Some statistical issues in the comparison of speech recognition algorithms. *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535 vol.1, May 1989.

[GLAJ99] Jean-Luc Gauvain, Lori Lamel, Gilles Adda, and Michèle Jardino. The LIMSI 1998 hub-4e transcription system. In *Proc. of the DARPA Broadcast News Workshop*, pages 99–104, Feb 1999.

[HBD$^+$07] Thomas Hain, Lukas Burget, John Dines, Giulia Garau, Martin Karafiat, Mike Lincoln, Jithendra Vepa, and Vincent Wan. The AMI system for the transcription of speech in meetings. In *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, USA, 2007.

[HCC$^+$06] A.G. Hauptmann, M.-Y. Chen, M. Christel1, D. Das, W.-H. Lin, R. Yan, J. Yang, G. Backfried, and X. Wu. Multi-lingual broadcast news retrieval. In *NIST TRECVID Workshop*, Gaithersburg, MD, November 2006.

[HCS99] Juan M. Huerta, Stanley Chen, and Richard M. Stern. The 1998 carnegie mellon university sphinx-3 spanish broadcast news transcription system. In *Proceedings of the 1999 DARPA Broadcast News Workshop*, 1999.

[Her90] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *Acoustical Society of America*, 87(4):1738–1752, 1990.

[HJT$^+$98] T. Hain, S.E. Johnson, A. Tuerk, P.C. Woodland, and .S.J. Young. Segment generation and clustering in the HTK broadcast news transcription system. In *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, pages 133–137, Virginia, USA, 1998.

[HMV+07] Jing Huang, Etienne Marcheret, Karthik Visweswariah, Vit Libal, and Gerasimos Potamianos. The IBM rich transcription 2007 speech-to-text systems for lecture meetings. In *Proceedings of the NIST Rich Transcription 2007 Spring Meeting Recognition Evaluation, RT07s*, Baltimore, USA, May 2007.

[HOdJ05] Marijn Huijbregts, Roeland Ordelman, and Franciska de Jong. A spoken document retrieval application in the oral history domain. In *Proceedings of 10th international conference Speech and Computer, Patras, Greece (SPECOM 2005)*, pages 699–702. University of Patras, Wire Communications Laboratory Moscow State Linguistics University, 2005. ISBN=5-7452-0110-x.

[HOdJ07] Marijn Huijbregts, Roeland Ordelman, and Franciska de Jong. Annotation of heterogeneous multimedia content using automatic speech recognition. In *Proceedings of the second international conference on Semantics And digital Media Technologies (SAMT)*, Lecture Notes in Computer Science, Berlin, December 2007. Springer Verlag.

[HOdJ08] Marijn Huijbregts, Roeland Ordelman, and Franciska de Jong. Fast N-Gram language model look-ahead for decoders with static pronunciation prefix trees. In *proceedings of Interspeech*, Brisbane, Australia, September 2008.

[HOvH01] Marijn Huijbregts, Roeland Ordelman, and Arjan van Hessen. Prosody based boundary detection. Technical report, University of Twente, Parlevink Group, 2001.

[HW07] Marijn Huijbregts and Chuck Wooters. The blame game: Performance analysis of speaker diarization system components. In *proceedings of Interspeech*, Antwerp, Belgium, August 2007.

[HWO07] Marijn Huijbregts, Chuck Wooters, and Roeland Ordelman. Filtering the unknown: Speech activity detection in heterogeneous video collections. In *proceedings of Interspeech*, Antwerp, Belgium, August 2007.

[ID71] M. Ito and R. Donaldson. Zero-crossing measurements for analysis and recognition of speech sounds. *Audio and Electroacoustics, IEEE Transactions on*, 19(3):235–242, Sep 1971.

[IFM+06] Dan Istrate, Corinne Fredouille, Sylvain Meignier, Laurent Besacier, and Jean François Bonastre. NIST RT05S evaluation: Pre-processing techniques and speaker diarization on multiple microphone meetings. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, Berlin, February 2006. Springer Verlag.

[Jel97] Frederick Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts, London, England, 1997.

[JH95] Jean-Claude Junqua and Jean-Paul Haton. *Robustness in Automatic Speech Recognition: Fundamentals and Applications.* Kluwer Academic Publishers, Norwell, MA, USA, 1995.

[JLSW04] Qin Jin, Kornel Laskowski, Tanja Schultz, and Alex Waibel. Speaker segmentation and clustering in meetings. proceedings of the NIST RT04s Evaluation Workshop, Montreal, Canada, May 2004.

[JM00] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice-Hall, New Jersey, 2000.

[JY94] D. A. James and S. J. Young. A fast lattice-based approach to vocabulary independent wordspotting. In *Proc. ICASSP '94*, pages I–377–I–380, Adelaide, Austrailia, 1994.

[KEH$^+$03] D. Kim, G. Evermann, T. Hain, D. Mrva, S. Tranter, L. Wang, and P. Woodland. Recent advances in broadcast news transcription. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 105–110., St. Thomas, U.S. Virgin Islands, 2003.

[KOIS06] W. Kraaij, P. Over, T. Ianeva, and A.F Smeaton. TRECVID 2006 - an introduction. In *proceedings of TRECVID 2006.* NIST, USA, 2006.

[KSWW00] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel. Strategies for automatic segmentation of audio data. In *proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1423–1426, Istanbul, Turkey, 2000.

[KvL07] Judith Kessens and David van Leeuwen. N-best: The northern- and southern-dutch benchmark evaluation of speech recognition technology. In *Interspeech*, Antwerp, Belgium, August 2007.

[LBG$^+$07] L. Lamel, E. Bilinksi, J.L. Gauvain, G. Adda, C. Barras, and X. Zhu. The LIMSI rt07 lecture transcription system. In *Proceedings of the NIST Rich Transcription 2007 Spring Meeting Recognition Evaluation, RT07s*, Baltimore, USA, May 2007.

[Li08] Haizhou Li. Query-by-example spoken document retrieval. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[LLJ90] C.-H. Lee, C.-H. Lin, and B.-H. Juang. A study on speaker adaptation of continuous density hmm parameters. *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 145–148 vol.1, 3-6 Apr 1990.

[LW95] C. Leggetter and C. Woodland. Flexible speaker adaptation using maximum likelihood linear regression. In *proceedings of Eurospeech*, pages 1155–1158, 1995.

[LW06] Jia Li and James Z. Wang. Real-time computerized annotation of pictures. In *proceedings ACM Multimedia*, pages 911–920, New York, NY, USA, 2006.

[Met05] Florian Metze. *Articulatory Features for Conversational Speech Recognition.* PhD thesis, University of Fridericiana, Karlsruhe, Germany, December 2005.

[MFP$^+$04] Florian Metze, Christian Fügen, Yue Pan, Tanja Schultz, and Hua Yu. The ISL RT-04S meeting transcription system. proceedings of the NIST RT04s Evaluation Workshop, Montreal, Canada, May 2004.

[MGMMC04] H. Mueller, A. Geissbuhler, S. Marchand-Maillet, and P. Clough. Benchmarking image retrieval applications. In *proceedings of the Seventh International Conference on Visual Information Systems*, San Francisco, USA, September 2004.

[MMRS08] Jonathan Mamou, Yosi Mass, Bhuvana Ramabhadran, and Benjamin Sznajder. Combination of multiple speech transcription methods for vocabulary independent search. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[MNL05] Dominique Massonie, Pascal Nocera, and Georges Linares. Scalable language model look-ahead for lvcsr. In *proceedings Interspeech 2005*, pages 569–572, Lisbon, Portugal, 2005.

[Mol03] Sirko Molau. *Normalization in the Acoustic Feature Space for Improved Speech Recognition*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, Germany, February 2003.

[MS03] S. Matsoukas and R. Schwartz. Improved speaker adaptation using speaker dependent feature projections. In *proceedings of IEEE workshop on Automatic Speech Recognition and Understanding*, pages 273–278, St. Thomas, Virgin Islands, U.S., 2003.

[MW06] Nikki Mirghafori and Chuck Wooters. Nuts and flakes: A study of data characteristics in speaker diarization. Toulouse, France, May 2006.

[NAA$^+$04] L. Nguyen, S. Abdou, M. Afify, J. Makhoul, S. Matsoukas, R. Schwartz, B. Xiang, L. Lamel, J.L. Gauvain, G. Adda, H. Schwenk, and F. Lefevre. The 2004 BBN/LIMSI 10xRT english broadcast news transcription system. In *Proc. DARPA RT04*, Palisades NY, November 2004.

[NIS06] NIST. Rich Transcription 2006 Spring Meeting Recognition evaluation plan V2. In *Rich Transcription 2006 Meeting Recognition Workshop*, 2006.

[NIS07] NIST. Spring 2007 (RT-07) Rich Transcription Meeting Recognition Evaluation Plan. In *Rich Transcription 2007 Meeting Recognition Workshop*, 2007.

[Oar04] D.W. Oard. Transforming access to the spoken word. In *proceedings of the International Symposium on Large-Scale Knowledge Resources*. Tokyo Institute of Technology, March 2004.

[ONEC96] S. Ortmanns, H. Ney, A. Eiden, and N. Coenen. Look-ahead techniques for improved beam search. In *proceedings of the CRIM-FORWISS Workshop*, pages 10–22, Montreal, 1996.

[Oos00] N. Oostdijk. The Spoken Dutch Corpus. Overview and first evaluation. In M. Gravilidou, G. Carayannis, S. Markantonatou, S. Piperidis, and G. Stainhaouer, editors, *Second International Conference on Language Resources and Evaluation*, volume II, pages 887–894, 2000.

[Ord03] Roeland Ordelman. *Dutch Speech Recognition in Multimedia Information Retrieval*. PhD thesis, University of Twente, The Netherlands, October 2003.

[Pal03] David S. Pallett. A look at NIST's benchmark ASR tests: Past, present, and future. In *Proceedings of 2003 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2003.

[PFF90] D.S. Pallet, W.M. Fisher, and Jonathan G. Fiscus. Tools for the analysis of benchmark speech recognition tests. *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 97–100 vol.1, Apr 1990.

[PH03] B. Pellom and K. Hacioglu. Recent Improvements in the CU Sonic ASR system for Noisy Speech: The SPINE Task. In *Proc. ICASSP*, 2003.

[RCL91] R. C. Rose, E. I. Chang, and R. P. Lippmann. Techniques for information retrieval from voice messages. In *proceedings of the Acoustics, Speech, and Signal Processing (ICASSP)*, pages 317–320, Washington, DC, USA, 1991. IEEE Computer Society.

[RSB$^+$07] Elias Rentzeperis, Andreas Stergiou, Christos Boukis, Aristodemos Pnevmatikakis, and Lazaros C. Polymenakos. The 2006 athens information technology speech activity detection and speaker diarization systems. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, Berlin, January 2007. Springer Verlag.

[SAB$^+$07] Andreas Stolcke, Xavier Anguera, Kofi Boakye, Özgür Çetin, Adam Janin, Mathew Magimai-Doss, Chuck Wooters, and Jing Zheng. The SRI-ICSI spring 2007 meeting and lecture recognition system. In *Proceedings of the NIST Rich Transcription 2007 Spring Meeting Recognition Evaluation, RT07s*, Baltimore, USA, May 2007.

[SCD$^+$04] Richard Schwartz, Thomas Colthurst, Nicolae Duta, Herb Gish, Rukmini Iyer, Chia-Lin Kao, Daben Liu, Owen Kimball, J. Ma, John Makhoul, Spyros Matsoukas, Long Nguyen, Mohamed Noamany, Rohit Prasad, Bing Xiang, Dongxin Xu, Jean-Luc Gauvain, Lori Lamel, Holger Schwenk, Gilles Adda, and Langzhou Chen. Speech recognition in multiple languages and domains: The 2003 BBN/LIMSI EARS system. In *Proceedings of ICASSP*, Montreal, May 2004.

[Sch78] G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[SCK$^+$85] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *proceedings of ICASSP '85*, pages 1205–1208, 1985.

[SFB08] Igor Szöke, Michal Fapšo, and Lukáš Burget. Hybrid word-subword decoding for spoken term detection. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[SL98] K. Shinoda and Chin-Hui Lee. Unsupervised adaptation using structural bayes approach. In *proceedings of the IEEE international conference on acoustics, speech and signal processing*, pages 793–796 vol.2, 12-15 May 1998.

[SMFW02] Hagen Soltau, Florian Metze, Christian Fugen, and Alex Waibel. Efficient language model lookahead through polymorphic linguistic context assignment. 2002.

[SML00] O. Siohan, T. Myrvol, and C. Lee. Structural maximum a posteriori linear regression for fast hmm adaptation. In *proceedings of ISCA ITRW Automatic Speech Recognition: Challenges for the Millenium*, pages 120–127, 2000.

[SMQS98] A. Smeaton, M. Morony, G. Quinn, and R. Scaife. Taisceala i: Information retrieval from an archive of spoken radio news. In *In proceedings of the Second European Digital Libraries Conference*, 1998.

[SSB$^+$05] Igor Szöke, Petr Schwarz, Lukáš Burget, Michal Fapšo, Martin Karafiát, Jan Černocký, and Pavel Matějka. Comparison of keyword spotting approaches for informal continuous speech. In *proceedings Interspeech 2005*, pages 633–636, 2005.

[STN94] V. Steinbiss, B.-H. Tran, and H. Ney. Improvements in beam search. In *proceedings of International Conference on Spoken Language Processing*, pages 1355–1358, Yokohama, Japan, 1994.

[Sto02] Andreas Stolcke. SRILM – an extensible language modeling toolkit. In *International conference on spoken language processing*, 2002.

[vdWH08] Laurens van der Werff and Willemijn Heeren. Subword-based indexing for a minimal false positive rate, research proposal. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[VFM07] O. Vinyals, G. Friedland, and N. Mirghafori. Revisiting a basic function on current cpus: A fast logarithm implementation with adjustable accuracy. Technical report, International Computer Science Institute, 2007.

[vL06] David van Leeuwen. The TNO speaker diarization system for NIST RT05s meeting data. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, pages 440–449, Berlin, February 2006. Springer Verlag.

[vLH07] David van Leeuwen and Marijn Huijbregts. The AMI speaker diarization system for NIST RT06s meeting data. In *Machine Learning for Multimodal Interaction (MLMI)*, volume 4299 of *Lecture Notes in Computer Science*, pages 371–384, Berlin, October 2007. Springer Verlag.

[vLK07] David van Leeuwen and Matej Koneĉný. Progress in the AMIDA speaker diarization system for meeting data. In *Multimodal Technologies for Perception of Humans*, Lecture Notes in Computer Science, Berlin, 2007. Springer Verlag.

[VSW$^+$04] Anand Venkataraman, Andreas Stolcke, Wen Wang, Dimitra Vergyri, Venkata Ramana Rao Gadde, and Jing Zheng. SRI's 2004 broadcast news speech to text system. In *Proceedings of EARS Rich Transcription 2004 workshop*, Palisades, 2004.

[vV07] Pieter van Veelen. Clustered acoustic modelling. Master's thesis, University of Twente, 2007.

[WFPA04] Chuck Wooters, James Fung, Barbara Peskin, and Xavier Anguera. Towards robust speaker segmentation: The ICSI-SRI fall 2004 diarization system. In *Fall 2004 Rich Transcription Workshop (RT04)*, Palisades, NY, November 2004.

[WH08] Chuck Wooters and Marijn Huijbregts. The ICSI RT07s speaker diarization system. In *Multimodal Technologies for Perception of Humans*, Lecture Notes in Computer Science, Berlin, 2008. Springer Verlag.

[WLY$^+$99] Xintian Wu, Chaojun Liu, Yonghong Yan, Doughwa Kim, Seth Cameron, and Randy Parr. The 1998 OGI-Fonix broadcast news transcription system. In *Proceedings of the 1999 DARPA Broadcast News Workshop*, 1999.

[WMOP96] S. Wegmann, D. McAllaster, J. Orloff, and B. Peskin. Speaker normalization on conversational telephone speech. *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, 1:339–341, 7-10 May 1996.

[WN49] Wiener and Norbert. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* Wiley, 1949.

[WSK07] Matthias Wölfel, Sebastian Stüker, and Florian Kraft. The ISL RT-07 speech-to-text system. In *Proceedings of the NIST Rich Transcription 2007 Spring Meeting Recognition Evaluation, RT07s*, Baltimore, USA, May 2007.

[YEH$^+$95] Steve J. Young, Gunnar Evermann, Thomas Hain, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book.* 1995.

[YOW94] Steve J. Young, J. Odell, and P. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *proceedings of the ARPA Workshop on Human Language Technology*, pages 307–312, 1994.

[YRT89] Steve J. Young, N.H. Russell, and J.H.S Thornton. Token passing: A simple conceptual model for connected speech recognition systems. Technical report, Cambridge University Engineering Dept, 1989.

[YTS08] Roger Peng Yu, Kit Thambiratnam, and Frank Seide. Word-lattice based spoken-document indexing with standard text indexers. In *proceedings of the ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech*, Singapore, 2008.

[ZBLG07] Xuan Zhu, Claude Barras, Lori Lamel, and Jean-Luc Gauvain. Speaker diarization: From broadcast news to lectures. In *Machine Learning for Multimodal Interaction (MLMI)*, Lecture Notes in Computer Science, Berlin, January 2007. Springer Verlag.

[ZWG99] P. Zhan, S. Wegmann, and L. Gillick. Dragon systems' 1998 broadcast news transcription system for mandarin. In *Proceedings of the 1999 DARPA Broadcast News Workshop*, 1999.

SUMMARY

In this thesis, research on large vocabulary continuous speech recognition for unknown audio conditions is presented. For automatic speech recognition systems based on statistical methods, it is important that the conditions of the audio used for training the statistical models match the conditions of the audio to be processed. Any mismatch will decrease the accuracy of the recognition. If it is unpredictable what kind of data can be expected, or in other words if the conditions of the audio to be processed are unknown, it is impossible to tune the models. If the material consists of 'surprise data' the output of the system is likely to be poor. In this thesis methods are presented for which no external training data is required for training models. These novel methods have been implemented in a large vocabulary continuous speech recognition system called SHoUT. This system consists of three subsystems: speech/non-speech classification, speaker diarization and automatic speech recognition.

The speech/non-speech classification subsystem separates speech from silence and unknown audible non-speech events. The type of non-speech present in audio recordings can vary from paper shuffling in recordings of meetings to sound effects in television shows. Because it is unknown what type of non-speech needs to be detected, it is not possible to train high quality statistical models for each type of non-speech sound. The speech/non-speech classification subsystem, also called the speech activity detection subsystem, does not attempt to classify all audible non-speech in a single run. Instead, first a bootstrap speech/silence classification is obtained using a standard speech activity component. Next, the models for speech, silence and audible non-speech are trained on the target audio using the bootstrap classification. This approach makes it possible to classify speech and non-speech with high accuracy, without the need to know what kinds of sound are present in the audio recording.

Once all non-speech is filtered out of the audio, it is the task of the speaker diarization subsystem to determine how many speakers occur in the recording and exactly when they are speaking. The speaker diarization subsystem applies agglomerative clustering to create clusters of speech fragments for each speaker in the recording. First, statistical speaker models are created on random chunks of the recording and by iteratively realigning the data, retraining the models and merging models that

represent the same speaker, accurate speaker models are obtained for speaker clustering. This method does not require any statistical models developed on a training set, which makes the diarization subsystem insensitive for variation in audio conditions. Unfortunately, because the algorithm is of complexity $O(n^3)$, this clustering method is slow for long recordings. Two variations of the subsystem are presented that reduce the needed computational effort, so that the subsystem is applicable for long audio recordings as well.

The automatic speech recognition subsystem developed for this research, is based on Viterbi decoding on a fixed pronunciation prefix tree. Using the fixed tree, a flexible modular decoder could be developed, but it was not straightforward to apply full language model look-ahead efficiently. In this thesis a novel method is discussed that makes it possible to apply language model look-ahead effectively on the fixed tree. Also, to obtain higher speech recognition accuracy on audio with unknown acoustical conditions, a selection from the numerous known methods that exist for robust automatic speech recognition is applied and evaluated in this thesis.

The three individual subsystems as well as the entire system have been successfully evaluated on three international benchmarks. The diarization subsystem has been evaluated at the NIST RT06s benchmark and the speech activity detection subsystem has been tested at RT07s. The entire system was evaluated at N-Best, the first automatic speech recognition benchmark for Dutch.

SAMENVATTING

In dit proefschrift wordt onderzoek gepresenteerd over continue automatische spraakherkenning met groot vocabulair. Voor automatische spraakherkenningssystemen die zijn gebaseerd op statistische methoden is het belangrijk dat de condities van de audio die worden gebruikt voor het trainen van de statistische modellen overeenkomen met de condities van de audio die verwerkt moeten worden. Elk verschil zal bijdragen tot een minder accurate herkenning. Als vooraf niet te voorspellen is wat voor soort data verwerkt moeten worden, of in andere woorden als de condities van de audio onbekend zijn, dan is het onmogelijk om de statistische modellen optimaal af te stemmen. Als het materiaal bestaat uit 'verrassingsdata' dan is de uitvoer van het systeem hoogst waarschijnlijk erg slecht. In dit proefschrift worden nieuwe methoden gepresenteerd waarvoor geen externe trainingsdata nodig zijn om modellen te trainen. Deze methoden zijn geïmplementeerd in het spraakherkenningssysteem SHoUT. Dit systeem bestaat uit drie subsystemen: spraak/geen-spraak classificatie, spreker-clustering en automatische spraakherkenning.

Het spraak/geen-spraak classificatie-subsysteem scheidt spraak van stilte en andere onbekende geluiden. Het soort geluiden dat voorkomt in audio-opnames kan variëren van papiergeritsel in opnames van vergaderingen tot geluidseffecten in televisieprogramma's. Omdat het onbekend is wat voor soort geluid gedetecteerd moet worden, is het niet mogelijk om hiervoor hoge kwaliteit statistische modellen te trainen. Het spraak/geen-spraak classificatie subsysteem, probeert niet in één keer alle geluiden te classificeren. In plaats daarvan wordt eerst een bootstrap spraak/stilte classificatie uitgevoerd met een standaard component. Door gebruik te maken van deze bootstrap-segmentatie kunnen modellen voor spraak, stilte en geluid getraind worden. Deze aanpak maakt het mogelijk om met hoge nauwkeurigheid spraak/geen-spraak te classificeren zonder de noodzaak om te weten wat voor soort geluiden aanwezig zijn in de opname.

Nadat de spraak van alle geluiden is gescheiden, is het de taak van het spreker-clustering subsysteem, ook wel 'diarization' subsysteem genoemd, om te bepalen hoeveel sprekers voorkomen in de opname en wanneer deze sprekers precies praten. Het 'diarization' subsysteem gebruikt een iteratieve methode om clusters van spraak

te genereren voor elke spreker in de opname. Eerst wordt een relatief groot aantal statistische sprekermodellen gemaakt op willekeurige fragmenten van de opname. Door daarna iteratief de data opnieuw op te lijnen, de modellen opnieuw te trainen en de modellen die dezelfde spreker representeren samen te voegen, worden steeds nauwkeurigere sprekermodellen gemaakt. Deze methode maakt geen gebruik van statistische modellen die getraind zijn op een trainingsset en hierdoor is het diarization subsysteem ongevoelig voor variaties in audiocondities. Helaas is de complexiteit van het algorithme $O(n^3)$ waardoor de clusteringsmethode niet erg geschikt is voor het verwerken van lange opnames. Twee variaties van het subsysteem worden gepresenteerd die minder computationele eisen stellen, zodat het subsysteem ook ingezet kan worden voor lange audiobestanden.

Het automatische spraakherkennings-subsysteem dat is ontwikkeld voor dit onderzoek, is gebaseerd op Viterbi decodering op een statische uitspraakboom (fixed pronunciation prefix tree). Door het gebruik van statische uitspraakbomen is het mogelijk om een flexibel en modulair opgebouwde decoder te ontwikkelen, maar door de statische uitspraakbomen is het niet eenvoudig om volledige taalmodel integratie (full language model look-ahead) toe te passen. In dit proefschrift wordt een nieuwe methode besproken dat het mogelijk maakt om volledige taalmodel integratie toch effectief toe te passen in de decoder. Ook wordt in dit proefschrift een selectie uit de ontelbare methoden voor robuste spraakherkenning toegepast en geëvalueerd.

De drie individuele subsystemen en het totale systeem zijn met succes geëvalueerd op drie internationale benchmarks. Het diarization subsysteem is getest op de NIST RT06s benchmark en het spraak/geen-spraak classificatie subsysteem is getest op RT07s. Het totale systeem is geëvalueerd op N-Best, de eerste automatische spraakherkenningsbenchmark voor het Nederlands.

SIKS SERIES

Since 1998, all dissertations written by Ph.D. students who have conducted their research under auspices of a senior research fellow of the SIKS research school are published in the SIKS Dissertation Series. This thesis is the 190th in the series.

2008-26 Marijn Huijbregts (UT), *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*
2008-25 Geert Jonker (UU), *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*
2008-24 Zharko Aleksovski (VU), *Using background knowledge in ontology matching*
2008-23 Stefan Visscher (UU), *Bayesian network models for the management of ventilator-associated pneumonia*
2008-22 Henk Koning (UU), *Communication of IT-Architecture*
2008-21 Krisztian Balog (UVA), *People Search in the Enterprise*
2008-20 Rex Arendsen (UVA), *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven.*
2008-19 Henning Rode (UT), *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*
2008-18 Guido de Croon (UM), *Adaptive Active Vision*
2008-17 Martin Op 't Land (TUD), *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
2008-16 Henriëtte van Vugt (VU), *Embodied agents from a user's perspective*
2008-15 Martijn van Otterlo (UT), *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.*
2008-14 Arthur van Bunningen (UT), *Context-Aware Querying; Better Answers with Less Effort*
2008-13 Caterina Carraciolo (UVA), *Topic Driven Access to Scientific Handbooks*
2008-12 József Farkas (RUN), *A Semiotically Oriented Cognitive Model of Knowledge Representation*
2008-11 Vera Kartseva (VU), *Designing Controls for Network Organizations: A Value-Based Approach*
2008-10 Wauter Bosma (UT), *Discourse oriented summarization*
2008-09 Christof van Nimwegen (UU), *The paradox of the guided user: assistance can be counter-effective*
2008-08 Janneke Bolt (UU), *Bayesian Networks: Aspects of Approximate Inference*
2008-07 Peter van Rosmalen (OU), *Supporting the tutor in the design and support of adaptive e-learning*
2008-06 Arjen Hommersom (RUN), *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
2008-05 Bela Mutschler (UT), *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
2008-04 Ander de Keijzer (UT), *Management of Uncertain Data – towards unattended integration*
2008-03 Vera Hollink (UVA), *Optimizing hierarchical menus: a usage-based approach*
2008-02 Alexei Sharpanskykh (VU), *On Computer-Aided Methods for Modeling and Analysis of Organizations*
2008-01 Katalin Boer-Sorbán (EUR), *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2007-25 Joost Schalken (VU), *Empirical Investigations in Software Process Improvement*
2007-24 Georgina Ramírez Camps (CWI), *Structural Features in XML Retrieval*
2007-23 Peter Barna (TUE), *Specification of Application Logic in Web Information Systems*
2007-22 Zlatko Zlatev (UT), *Goal-oriented design of value and process models from patterns*
2007-21 Karianne Vermaas (UU), *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
2007-20 Slinger Jansen (UU), *Customer Configuration Updating in a Software Supply Network*
2007-19 David Levy (UM), *Intimate relationships with artificial partners*

2007-18 Bart Orriëns (UvT), *On the development an management of adaptive business collaborations*
2007-17 Theodore Charitos (UU), *Reasoning with Dynamic Networks in Practice*
2007-16 Davide Grossi (UU), *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
2007-15 Joyca Lacroix (UM), *NIM: a Situated Computational Memory Model*
2007-14 Niek Bergboer (UM), *Context-Based Image Analysis*
2007-13 Rutger Rienks (UT), *Meetings in Smart Environments; Implications of Progressing Technology*
2007-12 Marcel van Gerven (RUN), *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
2007-11 Natalia Stash (TUE), *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
2007-10 Huib Aldewereld (UU), *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
2007-09 David Mobach (VU), *Agent-Based Mediated Service Negotiation*
2007-08 Mark Hoogendoorn (VU), *Modeling of Change in Multi-Agent Organizations*
2007-07 Nataša Jovanović (UT), *To Whom It May Concern – Addressee Identification in Face-to-Face Meetings*
2007-06 Gilad Mishne (UVA), *Applied Text Analytics for Blogs*
2007-05 Bart Schermer (UL), *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
2007-04 Jurriaan van Diggelen (UU), *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
2007-03 Peter Mika (VU), *Social Networks and the Semantic Web*
2007-02 Wouter Teepe (RUG), *Reconciling Information Exchange and Confidentiality: A Formal Approach*
2007-01 Kees Leune (UvT), *Access Control and Service-Oriented Architectures*
2006-28 Börkur Sigurbjörnsson (UVA), *Focused Information Access using XML Element Retrieval*
2006-27 Stefano Bocconi (CWI), *Vox Populi: generating video documentaries from semantically annotated media repositories*
2006-26 Vojkan Mihajlović (UT), *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
2006-25 Madalina Drugan (UU), *Conditional log-likelihood MDL and Evolutionary MCMC*
2006-24 Laura Hollink (VU), *Semantic Annotation for Retrieval of Visual Resources*
2006-23 Ion Juvina (UU), *Development of Cognitive Model for Navigating on the Web*
2006-22 Paul de Vrieze (RUN), *Fundaments of Adaptive Personalisation*
2006-21 Bas van Gils (RUN), *Aptness on the Web*
2006-20 Marina Velikova (UvT), *Monotone models for prediction in data mining*
2006-19 Birna van Riemsdijk (UU), *Cognitive Agent Programming: A Semantic Approach*
2006-18 Valentin Zhizhkun (UVA), *Graph transformation for Natural Language Processing*
2006-17 Stacey Nagata (UU), *User Assistance for Multitasking with Interruptions on a Mobile Device*
2006-16 Carsten Riggelsen (UU), *Approximation Methods for Efficient Learning of Bayesian Networks*
2006-15 Rainer Malik (UU), *CONAN: Text Mining in the Biomedical Domain*
2006-14 Johan Hoorn (VU), *Software Requirements: Update, Upgrade, Redesign – towards a Theory of Requirements Change*
2006-13 Henk-Jan Lebbink (UU), *Dialogue and Decision Games for Information Exchanging Agents*
2006-12 Bert Bongers (VU), *Interactivation – Towards an e-cology of people, our technological environment, and the arts*
2006-11 Joeri van Ruth (UT), *Flattening Queries over Nested Data Types*
2006-10 Ronny Siebes (VU), *Semantic Routing in Peer-to-Peer Systems*
2006-09 Mohamed Wahdan (UM), *Automatic Formulation of the Auditor's Opinion*
2006-08 Eelco Herder (UT), *Forward, Back and Home Again – Analyzing User Behavior on the Web*
2006-07 Marko Smiljanic (UT), *XML schema matching – balancing efficiency and effectiveness by means of clustering*
2006-06 Ziv Baida (VU), *Software-aided Service Bundling – Intelligent Methods & Tools for Graphical Service Modeling*
2006-05 Cees Pierik (UU), *Validation Techniques for Object-Oriented Proof Outlines*
2006-04 Marta Sabou (VU), *Building Web Service Ontologies*
2006-03 Noor Christoph (UVA), *The role of metacognitive skills in learning to solve problems*
2006-02 Cristina Chisalita (VU), *Contextual issues in the design and use of information technology in organizations*
2006-01 Samuil Angelov (TUE), *Foundations of B2B Electronic Contracting*
2005-21 Wijnand Derks (UT), *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*
2005-20 Cristina Coteanu (UL), *Cyber Consumer Law, State of the Art and Perspectives*
2005-19 Michel van Dartel (UM), *Situated Representation*
2005-18 Danielle Sent (UU), *Test-selection strategies for probabilistic networks*
2005-17 Boris Shishkov (TUD), *Software Specification Based on Re-usable Business Components*
2005-16 Joris Graaumans (UU), *Usability of XML Query Languages*
2005-15 Tibor Bosse (VU), *Analysis of the Dynamics of Cognitive Processes*
2005-14 Borys Omelayenko (VU), *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
2005-13 Fred Hamburg (UL), *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
2005-12 Csaba Boer (EUR), *Distributed Simulation in Industry*
2005-11 Elth Ogston (VU), *Agent Based Matchmaking and Clustering – A Decentralized Approach to Search*
2005-10 Anders Bouwer (UVA), *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
2005-09 Jeen Broekstra (VU), *Storage, Querying and Inferencing for Semantic Web Languages*

2005-08 Richard Vdovjak (TUE), *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
2005-07 Flavius Frasincar (TUE), *Hypermedia Presentation Generation for Semantic Web Information Systems*
2005-06 Pieter Spronck (UM), *Adaptive Game AI*
2005-05 Gabriel Infante-Lopez (UVA), *Two-Level Probabilistic Grammars for Natural Language Parsing*
2005-04 Nirvana Meratnia (UT), *Towards Database Support for Moving Object data*
2005-03 Franc Grootjen (RUN), *A Pragmatic Approach to the Conceptualisation of Language*
2005-02 Erik van der Werf (UM)), *AI techniques for the game of Go*
2005-01 Floor Verdenius (UVA), *Methodological Aspects of Designing Induction-Based Applications*
2004-20 Madelon Evers (Nyenrode), *Learning from Design: facilitating multidisciplinary design teams*
2004-19 Thijs Westerveld (UT), *Using generative probabilistic models for multimedia retrieval*
2004-18 Vania Bessa Machado (UvA), *Supporting the Construction of Qualitative Knowledge Models*
2004-17 Mark Winands (UM), *Informed Search in Complex Games*
2004-16 Federico Divina (VU), *Hybrid Genetic Relational Search for Inductive Learning*
2004-15 Arno Knobbe (UU), *Multi-Relational Data Mining*
2004-14 Paul Harrenstein (UU), *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
2004-13 Wojciech Jamroga (UT), *Using Multiple Models of Reality: On Agents who Know how to Play*
2004-12 The Duy Bui (UT), *Creating emotions and facial expressions for embodied agents*
2004-11 Michel Klein (VU), *Change Management for Distributed Ontologies*
2004-10 Suzanne Kabel (UVA), *Knowledge-rich indexing of learning-objects*
2004-09 Martin Caminada (VU), *For the Sake of the Argument; explorations into argument-based reasoning*
2004-08 Joop Verbeek (UM), *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politile gegevensuitwisseling en digitale expertise*
2004-07 Elise Boltjes (UM), *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
2004-06 Bart-Jan Hommes (TUD), *The Evaluation of Business Process Modeling Techniques*
2004-05 Viara Popova (EUR), *Knowledge discovery and monotonicity*
2004-04 Chris van Aart (UVA), *Organizational Principles for Multi-Agent Architectures*
2004-03 Perry Groot (VU), *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
2004-02 Lai Xu (UvT), *Monitoring Multi-party Contracts for E-business*
2004-01 Virginia Dignum (UU), *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2003-18 Levente Kocsis (UM), *Learning Search Decisions*
2003-17 David Jansen (UT), *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
2003-16 Menzo Windhouwer (CWI), *Feature Grammar Systems – Incremental Maintenance of Indexes to Digital Media Warehouses*
2003-15 Mathijs de Weerdt (TUD), *Plan Merging in Multi-Agent Systems*
2003-14 Stijn Hoppenbrouwers (KUN), *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
2003-13 Jeroen Donkers (UM), *Nosce Hostem – Searching with Opponent Models*
2003-12 Roeland Ordelman (UT), *Dutch speech recognition in multimedia information retrieval*
2003-11 Simon Keizer (UT), *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
2003-10 Andreas Lincke (UvT), *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
2003-09 Rens Kortmann (UM), *The resolution of visually guided behaviour*
2003-08 Yongping Ran (UM), *Repair Based Scheduling*
2003-07 Machiel Jansen (UvA), *Formal Explorations of Knowledge Intensive Tasks*
2003-06 Boris van Schooten (UT), *Development and specification of virtual environments*
2003-05 Jos Lehmann (UVA), *Causation in Artificial Intelligence and Law – A modelling approach*
2003-04 Milan Petković (UT), *Content-Based Video Retrieval Supported by Database Technology*
2003-03 Martijn Schuemie (TUD), *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
2003-02 Jan Broersen (VU), *Modal Action Logics for Reasoning About Reactive Systems*
2003-01 Heiner Stuckenschmidt (VU), *Ontology-Based Information Sharing in Weakly Structured Environments*
2002-17 Stefan Manegold (UVA), *Understanding, Modeling, and Improving Main-Memory Database Performance*
2002-16 Pieter van Langen (VU), *The Anatomy of Design: Foundations, Models and Applications*
2002-15 Rik Eshuis (UT), *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
2002-14 Wieke de Vries (UU), *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
2002-13 Hongjing Wu (TUE), *A Reference Architecture for Adaptive Hypermedia Applications*
2002-12 Albrecht Schmidt (Uva), *Processing XML in Database Systems*
2002-11 Wouter C.A. Wijngaards (VU), *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
2002-10 Brian Sheppard (UM), *Towards Perfect Play of Scrabble*
2002-09 Willem-Jan van den Heuvel (KUB), *Integrating Modern Business Applications with Objectified Legacy Systems*
2002-08 Jaap Gordijn (VU), *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*
2002-07 Peter Boncz (CWI), *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*
2002-06 Laurens Mommers (UL), *Applied legal epistemology; Building a knowledge-based ontology of the legal domain*
2002-05 Radu Serban (VU), *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents*
2002-04 Juan Roberto Castelo Valdueza (UU), *The Discrete Acyclic Digraph Markov Model in Data Mining*

2002-03 Henk Ernst Blok (UT), *Database Optimization Aspects for Information Retrieval*
2002-02 Roelof van Zwol (UT), *Modelling and searching web-based document collections*
2002-01 Nico Lassing (VU), *Architecture-Level Modifiability Analysis*
2001-11 Tom M. van Engers (VUA), *Knowledge Management: The Role of Mental Models in Business Systems Design*
2001-10 Maarten Sierhuis (UvA), *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*
2001-09 Pieter Jan 't Hoen (RUL), *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*
2001-08 Pascal van Eck (VU), *A Compositional Semantic Structure for Multi-Agent Systems Dynamics.*
2001-07 Bastiaan Schonhage (VU), *Diva: Architectural Perspectives on Information Visualization*
2001-06 Martijn van Welie (VU), *Task-based User Interface Design*
2001-05 Jacco van Ossenbruggen (VU), *Processing Structured Hypermedia: A Matter of Style*
2001-04 Evgueni Smirnov (UM), *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
2001-03 Maarten van Someren (UvA), *Learning as problem solving*
2001-02 Koen Hindriks (UU), *Agent Programming Languages: Programming with Mental Models*
2001-01 Silja Renooij (UU), *Qualitative Approaches to Quantifying Probabilistic Networks*
2000-11 Jonas Karlsson (CWI), *Scalable Distributed Data Structures for Database Management*
2000-10 Niels Nes (CWI), *Image Database Management System Design Considerations, Algorithms and Architecture*
2000-09 Florian Waas (CWI), *Principles of Probabilistic Query Optimization*
2000-08 Veerle Coupé (EUR), *Sensitivity Analyis of Decision-Theoretic Networks*
2000-07 Niels Peek (UU), *Decision-theoretic Planning of Clinical Patient Management*
2000-06 Rogier van Eijk (UU), *Programming Languages for Agent Communication*
2000-05 Ruud van der Pol (UM), *Knowledge-based Query Formulation in Information Retrieval.*
2000-04 Geert de Haan (VU), *ETAG, A Formal Model of Competence Knowledge for User Interface Design*
2000-03 Carolien M.T. Metselaar (UVA), *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.*
2000-02 Koen Holtman (TUE), *Prototyping of CMS Storage Management*
2000-01 Frank Niessink (VU), *Perspectives on Improving Software Maintenance*
1999-08 Jacques H.J. Lenting (UM), *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.*
1999-07 David Spelt (UT), *Verification support for object database design*
1999-06 Niek J.E. Wijngaards (VU), *Re-design of compositional systems*
1999-05 Aldo de Moor (KUB), *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*
1999-04 Jacques Penders (UM), *The practical Art of Moving Physical Objects*
1999-03 Don Beal (UM), *The Nature of Minimax Search*
1999-02 Rob Potharst (EUR), *Classification using decision trees and neural nets*
1999-01 Mark Sloof (VU), *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*
1998-05 E.W.Oskamp (RUL), *Computerondersteuning bij Straftoemeting*
1998-04 Dennis Breuker (UM), *Memory versus Search in Games*
1998-03 Ans Steuten (TUD), *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*
1998-02 Floris Wiesman (UM), *Information Retrieval by Graphically Browsing Meta-Information*
1998-01 Johan van den Akker (CWI), *DEGAS – An Active, Temporal Database of Autonomous Objects*