

Speech-based Automatic Closed Caption Alignment

Jeroen Boogaard

February 2010



Speech-based Automatic Closed Caption Alignment

A thesis submitted in partial satisfaction of the requirements for the degree of Master of Science presented at Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science Man-Machine Interaction Group, Media & Knowledge Engineering, February 2010
ing. Jeroen Boogaard (1139711)
Jeroen.boogaard@gmail.com

Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Keywords: Closed Captioning, Subtitle Alignment, Natural Language Processing, Keyword Identification, Speech Recognition, Dynamic Programming



Thesis: "Speech-based Automatic Closed Caption Alignment"
J. A. Boogaard (1139711)

Graduation Committee

prof. drs. dr. L.J.M. Rothkrantz
dr. ir. P. Wiggers
ir. H. Geers
drs. ir. H. Jongbloed

Abstract

In the Netherlands, four million people watch television programs with closed captions because they are hearing impaired or non-native speakers. Closed captions contain Dutch speech transcriptions and non-speech sound descriptions and are displayed as subtitles. Due to government obligation, the number of television programs that must be closed-captioned will increase to at least 95% in 2011.

Closed caption alignment comprises the timing of the subtitles as closely as possible to the corresponding times of the video signal. Since alignment is a costly and labor intensive process demanding high quality outputs, an automated solution is desirable. This thesis addresses the application of automatic speech recognition to the task of off-line closed-captioning of television programs. The thesis focuses on the development of an automatic closed caption alignment system for TT888, a company that produces subtitles for Dutch-language television programs.

Investigation of related research, consulting professional editors and analyses of a variety of captioned television programs have contributed to the development of an automatic closed-captioning system named SETH (Speech Estimating Title Heuristics). The core of the system is an algorithm capable of matching manually produced captions with speech transcriptions produced by a large vocabulary speech recognizer. The architecture of SETH combines the benefits of modular programming and the pipes and filters architecture.

The best results are achieved when the speech is rather formal and non-spontaneous, pure Dutch pronounced by a native speaker and does not contain crosstalk nor background noise. Dissimilarities between the speech and captions are not a major problem as long as the captions include the most important words. The alignment algorithm is also robust to most of the insertions caused by music. Deviant language use, songs, spontaneous speech, strong regional accents are still a difficult job for the speech recognizer and hence a major problem in automatic closed caption alignment. Since there will always be broadcasts with poor speech quality, manual verification or adaptation of the subtitles remain necessary.

Preface

This thesis describes the research and development in partial satisfaction of the requirements for the degree of Master of Science presented at Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science Man-Machine Interaction Group. Both the literature survey [Boogaard07] and thesis project were performed at Dutchear, a small TNO spin-off in Speech Technology (www.dutchear.nl).

Project

During my graduation period at Dutchear, I have studied research papers, experimented with Speech Recognition software and conversion tools, and brainstormed about problems and possible solutions. All those things have contributed to the development of SETH: Speech Estimating Title Heuristics. SETH is a system of automatic subtitle preparation. The version of SETH that is discussed in this thesis, has been developed for and sold to the "Nederlandse Publieke Omroep (NPO) TT888". In the near future, a enhanced version of SETH will be developed for NPO. Currently, marked investigation is being performed to find new potential customers for SETH.

Acknowledgements

First of all, I would like to thank my supervisors L. Rothkrantz and H. Jongebroed. I would also like to thank E. Buitinga from TT888 for his information, ideas and test data. Thanks also to M. Simsic for his technical support. I would like to thank C. Quartel, L. Moreira Cardose and R. Zwanenburg for their contribution to the Graphical User Interface. Finally I would like to thank my girlfriend D. van Marion for her help with the layout.

1	Introduction	11
1.1	Motivation	11
1.2	Closed Caption Alignment	13
1.3	Automatic Closed Alignment	15
1.3.1	Automatic Speech Recognition	15
1.4	Problem definition	15
1.5	Complexity of the task	16
1.6	Methodology	17
1.7	Thesis Overview	17
I	Background	19
2	Automatic Speech Recognition	21
2.1	Feature Extraction	21
2.1.1	Frame Extraction	21
2.1.2	Spectral Features	22
2.1.3	Linear Predictive Coding	23
2.2	Speech Models	24
2.2.1	Lexicon	24
2.2.2	Acoustic Model	25
2.2.3	Language Model	28
2.2.4	Smoothing	28
2.3	Decoding	29
2.4	Speech Application Architectural Design	30
2.4.1	System Overview	30
2.4.2	Pipes and Filters Architecture	32
3	Related Work	33
3.1	Sentence Alignment in Machine Translation	33
3.2	Stream Alignment in Video Indexing	34
II	Analysis and Design	37
4	Data Examination	39
4.1	Boeken (VPRO)	39
4.2	Iedereen kan aquarelschilderen (Omroep MAX)	41
4.3	De vloer op (HOS)	43
4.4	NOVA (NPS)	45
4.5	't Vrije schaep (KRO)	48

4.6	Observations	49
5	Requirements	53
5.1	Speech Estimating Title Heuristics	53
5.2	Data Communication	54
5.3	Pre- and Post-processing	55
5.4	Model Training	55
5.5	Speech Decoding	56
5.6	Alignment	56
5.7	Logging, Error Handling and Updating	56
6	Design	57
6.1	System Overview	57
6.2	System States	58
6.3	Speech Models	61
6.3.1	Speech Model Pipeline	61
6.4	Alignment Algorithm	66
6.4.1	First-Pass Alignment	66
6.4.2	Second-Pass Alignment	70
III	Implementation	75
7	Components	77
7.1	Video2Audio Convertor	79
7.2	Speech Model Trainer	80
7.3	Speech Recognizer	80
7.4	String Matcher	80
7.5	Archiver	81
8	Modules	83
8.1	Perl	83
8.2	Import and Export Module	85
8.2.1	XML	86
8.3	Train Module	87
8.4	Test module	88
8.5	Align Module	90
8.6	Clean Module	90
8.7	Web Server	91
IV	Results, Conclusions and Recommendations	97
9	Results	99

10 Conclusions	101
10.1 Evaluation of Activities	101
10.2 Discussion	101
11 Recommendations for future work	103
Appendices	106
A Research Paper	106
B SONIC Lexicon	116
C Word Frequencies	123
D Output of Match	129
E Deamon Script	137
F Normalized Trainfile	138
G SONIC Configuration File	140
H Aligned Captions	142
Glossary	156
Bibliography	156

LIST OF FIGURES

1.1	Screen-shot News, with (right) and without (left) caption . . .	11
1.2	"Closed-captioning system with a re-speak method"	12
1.3	WinCaps	13
1.4	Garfield comic with text balloons ©Jim Davis	14
1.5	Clap-board to synchronize sound with video frames	14
2.1	A spectrum and its corresponding waveform	22
2.2	A spectrogram and corresponding waveform	23
2.3	A spectrogram of 'i', 'u' and 'a'	23
2.4	Overview of Automatic Speech Recognition	24
2.5	US English phonetic alphabet used by SONIC	25
2.6	Example of a Markov Model	26
2.7	Recognition of the word "speech"	27
2.8	Global design of a speech application	30
4.1	Wim Brands (left) and Jaap Jacobs in "Boeken"	39
4.2	Captions from "Boeken"	40
4.3	Video frame of animation synchronous to the music fragment	41
4.4	Captions from "Iedereen kan aquarelschilderen"	42
4.5	Speech signal containing silence (blue) and music (red) . . .	42
4.6	Video frame of "De vloer op"	43
4.7	Captions from "De vloer op"	43
4.8	Twan Huys reporting about Marc van Roosmalen	45
4.9	Captions from NOVA	46
4.10	Background music with (green) and without speech (red) . .	46
4.11	Screenshot of subtitled Brazilian speech	47
4.12	Cartoon requiring captions of non-verbal sounds	47
4.13	MarcMarie Huijbregts and Loes Luca in 't Vrije schaep	48
4.14	Piano sound with (green) and without (yellow) speech	48
4.15	Captions from 't Vrije schaep	49
5.1	Context of SETH	54
6.1	System overview of SETH	57
6.2	Entities used by SETH	59
6.3	States of SETH	60
6.4	Speech model pipeline	61
6.5	First Pass Alignment	72
6.6	ERD extended with entity "alternative caption"	73
6.7	Second Pass Alignment	74

7.1	Data flows SETH	78
7.2	Import raw audio in Audacity	79
7.3	Example of activation of CMU-CAM filters	80
7.4	Hakell implementation of equation 6.3	81
7.5	Archiving folder foo	81
8.1	Regular expression to parse time-coded words	84
8.2	FSM to parse time-coded words	84
8.3	API calls of FFmpeg and SoX respectively	85
8.4	stl file viewed in Hexedit	86
8.5	Shell script for the construction of a language model	87
8.6	Shell script for the construction of a lexicon	87
8.7	SONIC API calls	88
8.8	Two simultaneous SONIC instances	89
8.9	Form for submitting a stl-file and mpg-file	92
8.10	GUI for processes monitoring	93
8.11	Flowchart for determining the progress	94
8.12	Download-page for the new subtitle-file	95

LIST OF TABLES

1.1	Overview of project activities	17
3.1	Computation of $D(i, j)$	35
3.2	CC stream and its minimum distance variant	35
4.1	Overview of analyzed data	50
4.2	Overview of difficulties in the analyzed data	51
6.1	Specification speech model pipeline	62
6.2	Specification train-file filter chain	63
6.3	Specification word count filter chain	63
6.4	Specification lexicon filter chain	64
6.5	Specification language model filter chain	64
6.6	Origin off-the-shelf filters	64
6.7	Word counts of "Iedereen kan aquarelschilderen"	67
6.8	Alternative caption simulating insertions	70
6.9	Alternative captions and LD	71
9.1	Alignment results	99
9.2	Performance difference using universal speech models	99

1 INTRODUCTION

Closed captions contain Dutch speech transcriptions and non-speech sound descriptions and are displayed as subtitles. They are called *closed* since they remain separate from the video frames so that they may be optionally displayed as shown in figure 1.1. The opposite are *open* captions which are included as part of the video stream and are therefore visible to all viewers. Open captions usually provide translations in a different language.

This thesis is about the research and development involved in automatic closed caption alignment. Section 1.1 outlines why this is a relevant problem. Alignment means that the system is not supposed to *produce* the captions but to *synchronize* each caption with the soundtrack of the video signal. The alignment is currently being done manually as described in section 1.2. Section 1.3 discusses which Artificial Intelligence (AI) techniques are the most promising to automate the alignment process. The central problem is stated in section 1.4 while the expected challenges can be found in section 1.5. Finally, section 1.6 provides list of the research and development activities. A list of abbreviations can be found in the glossary.

1.1 Motivation

In the Netherlands, four million people watch television programs with closed captions because they are hearing impaired or non-native speakers. Closed captions can also be used to watch in noisy environments. Encoded as teletext data, the closed captions are transmitted as part of the broadcast signal in synchronization with the video signal. Due to government obligation, the number of television programs that must be closed-captioned will increase to at least 95% in 2011.

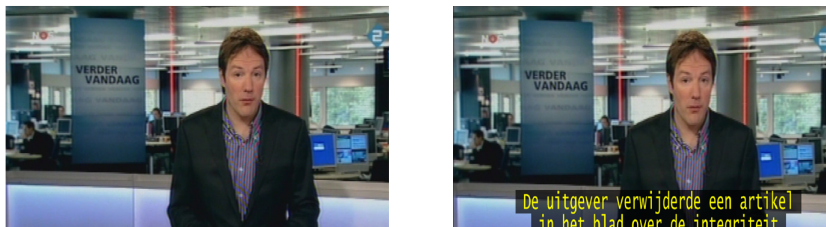


Figure 1.1: Screen-shot News, with (right) and without (left) caption

For some television programs like for example news, the auto-cue text can be used which can therefore be closed-captioned *offline*. An *online* method that is often used for live programs is re-speak. In the re-speak method, the re-speaker - a person other than the original speaker - carefully rephrases what he or she hears. The re-spoken utterances are fed to the speech recognizer that converts the speech to text. The text is confirmed and corrected by one or more correction operators. Finally, the resulting captions are broad-casted. Advantages of re-speak are the lack of background noise and overlapping speech, the ability to adapt the speech recognizer to the re-speaker and the possibility to supplement the speech by mentioning non-speech sounds like applause [Imai07]. The re-speak method is displayed in figure 1.2.

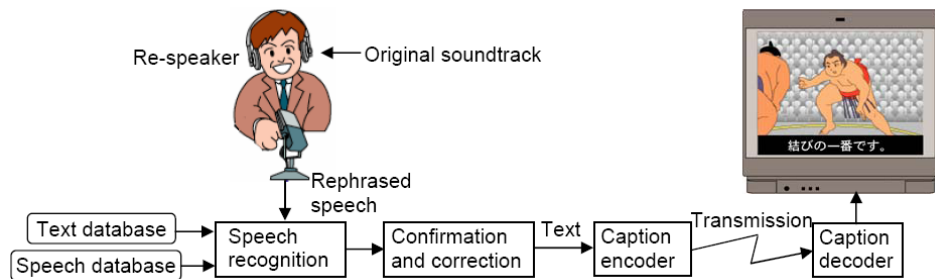


Figure 1.2: "Closed-captioning system with a re-speak method"

1.2 Closed Caption Alignment

The signal of a television broadcast contains a continuous speech signal which is not uniformly distributed over the video frames. Closed caption alignment is concerned with finding the time correspondence between speech utterances and video frames. The standard video field rate for PAL and SECAM television is 25 frames per second. Using subtitle software like WinCaps (see figure 1.3), the captions are manually placed along a time axis. Each speech utterance must be represented by a substituting caption that is attached to a number of video frames such that the *message* will still be interpreted correctly when the sound is muted.



Figure 1.3: WinCaps

To some degree, this problem is analogous to the problem of attaching text balloons in comics such that the information will be received in chronological order. As can be seen in figure 1.4¹ a comic has a number of key frames with captions attached to it. The same holds for subtitled videos but videos also have intermediate frames to render smooth motions. Rather than a picture of a speaking person, video streams allow people to follow the lip movements. Like comics, subtitled broadcasts also have the modalities image and text but are also accompanied by sound. In movie recordings, sound is being synchronized using a clap-board (see figure 1.5). If the sound is not synchronous with the video frames, speech will not correspond correctly with the lip movements of speakers which is rather annoying. Something similar, albeit to a lesser degree, holds for the synchronization of speech and closed captions. This is why *lip synchronization* is an important issue in closed caption alignment. Another issue that entails an accurate timing is that for utterances with short intermediate silences, the captions must smoothly proceed in each other to avoid flicker.

¹from <http://justmoney.wikispaces.com>



Figure 1.4: Garfield comic with text balloons ©Jim Davis



Figure 1.5: Clap-board to synchronize sound with video frames

1.3 Automatic Closed Alignment

To automate the subtitle preparation process, saving costs, some kind of AI-system is needed that is able to match the predefined utterances (captions) with spoken utterances (speech). In other words an automatic closed caption alignment system must be capable of speech estimation. Hence the most promising AI technique is based on Automatic Speech Recognition (ASR). This graduate project comprises the development of an automatic closed captioning system that can be used by Nederlandse Publieke Omroep (NPO) for their TT888 closed caption broadcasts.

1.3.1 Automatic Speech Recognition

ASR is the process of converting digitized speech into text. A speech recognizer is software that accepts recorded or streamed speech and returns transcriptions. A speech recognizer is not only able to determine *what* was being said also *when*. Therefore ASR can be used in reducing the amount of manual effort in the subtitling process and to satisfy the challenge of lip synchronization. For this we need the speech recognizer to retrieve the time-frames of the words in the captions. More about ASR can be found in chapter 2.

1.4 Problem definition

The problem of automatic closed caption alignment can be split into two main sub problems being the conversion of speech to text and the attachment of captions to video frames. The first problem raises questions like "What is the best way to use a speech recognizer for automatic closed caption alignment?" and "Which speech models can be general and which must be specifically trained for each new broadcast?". The second problem is concerned with questions like "What is the effect of recognition errors on the text-video synchronization?", "What is the effect of dissimilarities between the speech content and the captions?" and "How can these effects be minimized?".

1.5 Complexity of the task

Automatic closed-captioning is a complex task with many challenges. The performance highly relies on the accuracy of the Large-Vocabulary Continuous Speech Recognition (LVCSR) systems which have been traditionally developed for read speech with a close talking microphone. In Woodland et al. [Woodland97] et al. it is stated that broadcast speech is not homogeneous and includes a number of types for which current speech recognizers have large error rates. Broadcasts may contain speech from both native and non-native speakers or speech from different speech styles e.g. read, spontaneous and conversational. Due to contextual effects, words are sometimes pronounced differently in different contexts. Filler words like 'eh' and 'ehm' as well as breathe, lip-smacks and laughter also typically occur in spoken language. There is also variation in the quality of the audio signal which may suffer from a low bandwidth or too much background noise.

Some utterances are too long to fit in a two-lined subtitle and are therefore spanned over multiple subtitle screens (and thus multiple captions). Text reduction is applied when verbatim transcriptions result in more words than can be easily read within the display time. It primarily involves dropping less important words or phrases from a sentence into a more *condensed* format. Condensed captions preserve the most important words and maintain the original sentence structure as much as possible [Ahmer02]. An example of a condensed caption of the verbatim caption "Her stepmother did not like her one little bit" is "Her stepmother did not like her". Unfortunately, text reduction cause a mismatch between the lengths of the actual speech and the corresponding caption.

In contrast to open captions, closed captions contain descriptions of non-speech sound to ensure all relevant information is available to the hearing impaired viewer. The content non-speech captions are similar to the descriptions used in comics e.g. the word "foom" indicating the sound of a big flame in figure 1.4. Non-speech captions cause synchronization problems since it will be impossible to find the exact words of closed captions in the audio signal [Robert-Ribes98].

1.6 Methodology

The research will start with studying relevant literature about automatic closed caption alignment and related subjects. Next, video files and corresponding subtitle files will be collected for more insight into the problem domain. The data will be manually converted to formats that are suitable for experiments and inspection. A speech recognizer will be selected and initial speech models will be developed or reused from earlier projects to run ASR experiments. Speech models for speech transcription and an algorithm for matching the captions with the transcriptions will be designed and implemented. A suitable software architecture for the prototype will be designed and implemented. The system will be applied some test data and the results will be evaluated. An overview of the project activities is presented in 1.1.

Table 1.1: *Overview of project activities*

Activity	Goal
literature study	knowledge about solutions to related problems
data examination	insight into the problem and survey of pitfalls
analysis of system behavior	global design of prototype
architectural design	system subdivision into modules
model design	models for transcription and alignment
software implementation	working prototype
experiments	collection of test result data
evaluation	quality measurement and recommendations

1.7 Thesis Overview

Part I contains the necessary background information and a survey of related research. In part II the example data is analyzed, the requirements are stated and the design and methodology are presented. The subdivision of the system into functional components is explained in part III as well as the used tools and programming languages. An evaluation of the system can be found in part IV which also contains conclusions and recommendations.

Part I

Background

2 AUTOMATIC SPEECH RECOGNITION

Speech is an acoustic representation of a word or sequence of words produced by small movements of air particles. Humans perceive speech by converting spectral properties into an underlying word string and its associated meaning. The goal of ASR is to map the speech signal into a sequence of characters that forms the spoken sentence. The first step is to digitize (recorded, sampled, quantized) and store as a sequence of integer values (WAV). However, the speech one is dealing with in automatic closed caption alignment is already digitized albeit together with the camera signal. So this chapter starts by explaining how the digital speech signal can be represented by acoustic features in section 2.1. The result of the acoustic pre-processing is a series of observations for which the recognizer has to determine the most likely word sequence using so called speech models, defined in section 2.2. This problem is known as decoding and is the topic of section 2.3. Finally section 2.4 discusses the architectural design of speech applications.

2.1 Feature Extraction

A speech signal contains noise. This can be due to the environment and/or introduced by the speech recording equipment. Therefore a pre-processing step is necessary to store the relevant signal information into a compact, robust and reliable parametric representation. The aim of acoustic pre-processing is to separate acoustic events of interest.

2.1.1 Frame Extraction

During frame extraction the waveform is segmented into small overlapping time slices called frames. A frame has a window size of 25 milliseconds. From each second of speech, 100 frames are extracted each being stored in a vector. The resulting sequence of feature vectors is referred to as the observation sequence.

2.1.2 Spectral Features

The next step in the speech recognition process is to extract features i.e. signal measurements. A commonly used method is the Fourier transform. A Fourier transform is being applied to the speech waveform in order to analyze how the energy is distributed over different frequencies. The idea behind Fourier transform is that a complex wave can be regarded as a sum of many sine waves of different frequencies. The contributions of each frequency component can be represented by a spectrum as can be seen in figure see 2.1¹. A spectrum exposes the characteristic signature of a waveform.

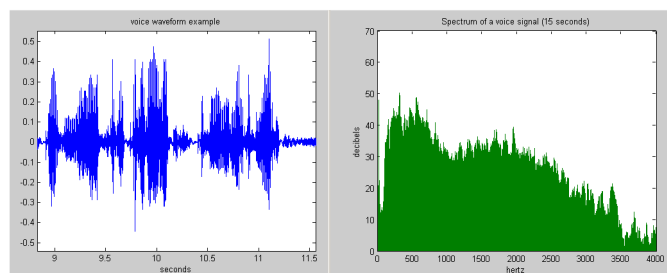


Figure 2.1: *A spectrum and its corresponding waveform*

While the spectrum shows the frequency components of a wave at one slice in time (a stochastic variable), a spectrogram illustrates the time-frequency relationship (a stochastic process). The spectrogram in figure 2.2 shows some red horizontal bars which indicate spectral peaks². These spectral peaks are called formants. The lower is the first and the higher is the second formant. Formants occur during voiced speech segments such as vowels and diphthongs and they reflect the different shapes of vocal tract [Rabiner93]. Figure 2.3 shows the spectrograms of three different vowels in American English³. The annotations 'F1' and 'F2' indicate the first and second formant respectively.

¹picture from <http://commons.wikimedia.org>

²picture from <http://www.wilhelm-kurz-software.de>

³picture from <http://commons.wikimedia.org>

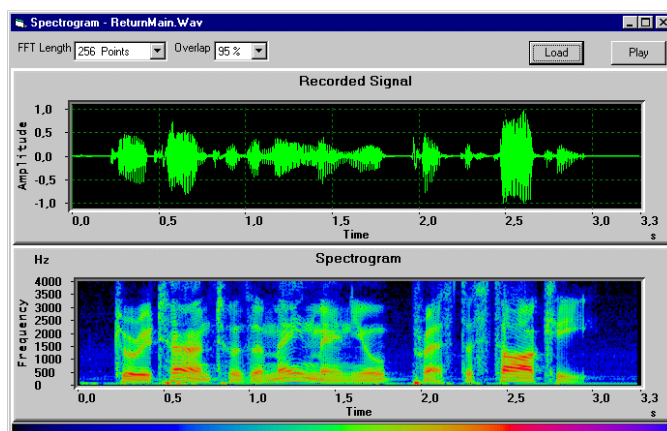


Figure 2.2: A spectrogram and corresponding waveform

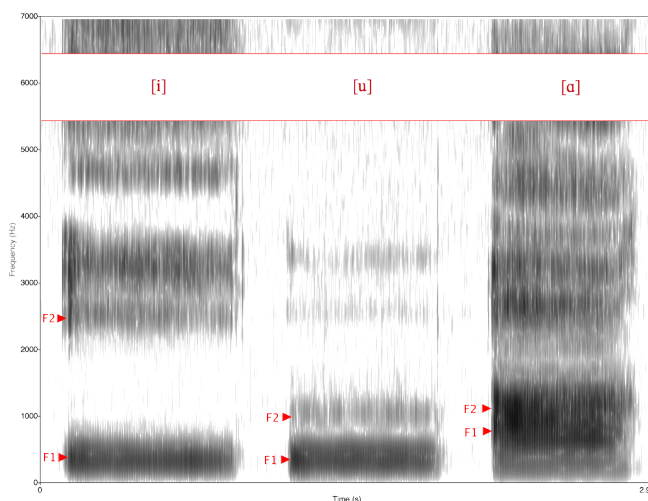


Figure 2.3: A spectrogram of 'i', 'u' and 'a'

2.1.3 Linear Predictive Coding

Most often a smoothed version of the spectrum is being used in which it is easier to see where the spectral peaks are [Heuvel63]. Such a spectrum can be obtained by applying Linear Predictive Coding (LPC). LPC is a technique to approximate a specific speech sample at the current time as a linear combination of past speech samples. The sum of squared differences between actual speech samples and corresponding linear predicted values is minimized resulting in a unique set of predictor coefficients. Since these coefficients characterize the shape of the vocal tract, they are useful features for speech recognition.

2.2 Speech Models

Figure 2.4 presents a schematic overview of ASR. The result of the feature extraction process, described in section 2.1, is a sequence of observations for which the best matching text has to be found. Speech models create a search space in which the most likely word sequence is to be found by optimization algorithms.

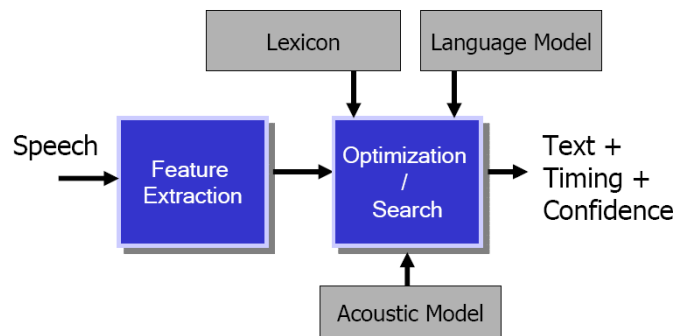


Figure 2.4: *Overview of Automatic Speech Recognition*

2.2.1 Lexicon

In a lexicon or pronunciation dictionary, words are represented by phonetic pronunciations [Jurafsky00]. A phoneme is an acoustic event with a duration varying from 10 to 100 ms. Phonemes are linguistically distinct sounds that can be divided into vowels, semivowels, and consonants which can be further subdivided into nasals, stops, fricatives and affricates. Every language has a unique phonetic alphabet. Figure 2.5 shows the US English phonemes used by the SONIC speech recognizer [SONIC]. An example of a complete SONIC lexicon can be found in appendix B.

Lexicons are used in Text-to-Speech (TTS) systems as well as in ASR (Speech-to-Text) systems as they present a mapping between speech units (phonemes) and text units (letters). In LVCSR, a lexicon describes the possible phoneme sequences. Lexica are very dynamic; if one is familiar with the phonetic alphabet, new words can easily be added.

Phone	Example	Phone	Example	Phone	Example	Phone	Example
AA	f <u>a</u> ther	DX	bu <u>t</u> ter	KD	ta <u>k</u>	GD	mu <u>g</u>
AE	ma <u>d</u>	DH	th <u>e</u> m	JH	Je <u>r</u> ry	SH	sh <u>o</u> w
AH	bu <u>t</u>	EH	be <u>d</u>	K	ki <u>t</u> ten	T	to <u>t</u>
AO	fo <u>r</u>	ER	bi <u>r</u> d	L	li <u>s</u> ten	TH	th <u>r</u> ead
AW	fr <u>o</u> wn	EY	sta <u>t</u> e	M	ma <u>n</u> ager	UH	ho <u>o</u> d
AX	alo <u>n</u> e	F	fr <u>i</u> end	N	na <u>n</u> cy	UW	mo <u>o</u> n
AXR	bu <u>t</u> ter	G	gr <u>o</u> wn	NG	fi <u>s</u> hing	V	ve <u>r</u> y
AY	hi <u>r</u> e	HH	ha <u>d</u>	OW	co <u>n</u> e	W	we <u>a</u> ther
B	bo <u>b</u>	IH	bi <u>t</u> ter	OY	bo <u>y</u>	Y	ye <u>l</u> low
CH	ch <u>u</u> rch	IX	ro <u>s</u> es	P	po <u>p</u>	Z	be <u>e</u> s
D	do <u>n</u> 't	IY	bea <u>t</u>	R	re <u>d</u>	ZH	mea <u>s</u> ure
PD	to <u>p</u>	BD	ta <u>b</u>	S	so <u>n</u> ic	SIL	sil <u>e</u> nce
TD	lo <u>t</u>	DD	ha <u>d</u>	TS	bi <u>t</u> s	br	<breath <u>e</u> >
ls	<lipsmack>	lg	<laughter>	ga	<garbage>		

Figure 2.5: US English phonetic alphabet used by SONIC

2.2.2 Acoustic Model

Given an observation sequence O , an acoustic model assign a score to each hypothesized word sequence W . A powerful statistical technique is the Hidden Markov Model (HMM) .

A Markov model is a Finite State Machine (FSM) in which the state transitions are associated with a probability distribution [Young97]. Since the states correspond to observations, a Markov model can be regarded as generator of random observation sequences. As can be seen in figure 2.6 each state in a Markov model generates exactly one observation⁴. A stationary observation sequence can be modeled by a Markov model if it satisfies the Markov assumption which states that the probability of a certain observation being generated is only dependent on the current state, not on any previously generated observations [Alphen92]. Stationary means that the state transition probabilities instantaneous i.e. independent of the actual time at which the transitions takes place. Although these assumptions are not sufficiently met by a speech signal, 'normal' Markov models cannot be used for speech recognition as the number of required states would be too large.

By relaxing the constraint that each state can only generate one observation, the number of states can be reduced drastically. This is exactly what happens in a HMM . For every state a probability distribution can be defined that specifies how likely each observation is to be generated in that particular state. The model is called *hidden* since the state sequence underlying an observation sequence is not visible any more.

⁴picture from <http://www.cis.hut.fi/Opinnot/T-61.184/>

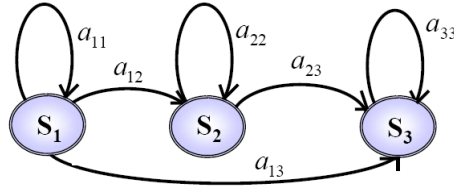


Figure 2.6: Example of a Markov Model

A HMM is defined by:

- * states $S = S_1, S_2, \dots, S_N$,
- * state transition probabilities a_{ij} ,
- * observation probabilities $A = [a_{ij}]$,
- * initial state probabilities $\pi = \pi_i$,
- * accepting states $S_a \subseteq S$.

A HMM automatically performs dynamic time warping and can deal with small distortions in a signal. A weakness of HMM's is duration modeling. Due to the amount of training data needed to cover the range of speaking variabilities, words cannot be directly modeled by HMM's for LVCSR. Phonemes, as defined in section 2.2.1, are often used since their linguistic level is low enough for adequate training [Wiggers08]. Phonemes are usually modeled by three pronunciation phases: on-glide, pure phoneme and off-glide. A single word is being recognized by passing through the states that compose the word as defined by the lexicon. An example of a recognition of the word "speech" by a left to right sub-word HMM is presented in figure 2.7⁵. Sub-word models account for different contexts but increase the amount of required training data or alternatively; a proper clustering technique. As will be described in section 2.2.3, the mapping between the sub-word HMM units and the word level Statistical Language Model (SLM) are provided by the pronunciation dictionary. It is allowed to define multiple pronunciations of the same (orthographic) word to account for different dialects. Note that this will cause multiple paths in the corresponding word level HMM.

⁵picture from <http://www.cis.hut.fi/Opinnot/T-61.184/>

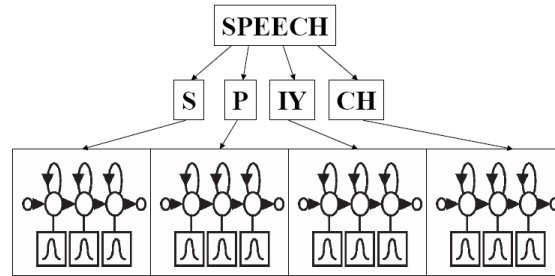


Figure 2.7: Recognition of the word "speech"

The problem to be solved for recognition can be stated as follows: "Given the model and observation sequence, what is the most likely underlying (hidden) state sequence?" This problem can be solved by the forward algorithm that calculates the probability for each possible path. The forward algorithm is computationally expensive. The Viterbi algorithm provides an efficient alternative by avoiding the necessity for examining all possible paths [Li00]. The parameters (probabilities) of an HMM are obtained by a process called training. For ASR, a corpus of annotated speech is used as training data. During training, the parameters are estimated from the data set such that they optimize an appropriate criterion like the Maximum Likelihood (ML). A popular technique for training an Acoustic Model (AM) for Speech Recognition is Baum-Welch re-estimation.

2.2.3 Language Model

The way in which words can be combined to form sentences is language and even context dependent. The task of a language model is to reduce the number of word choices during the recognition process. Some of the word sequences are impossible, others may be possible, and others may be very typical according to syntactic and semantic constraints. A SLM estimates the likeliness of word sequences by assigning a probability to every word sequence. Due to data sparseness it is impossible to model entire word sequences. Instead, n-gram models are used for the probability estimate [Gibbon97]. An n-gram model is a Markov model with an $n - 1$ restricted word-history. A trigram has a word history of (3-1) two words, a bigram takes (2-1) one preceding word into account, unigrams ($n = 1$) do not consider any preceding word. The probabilities are estimated by counting words (unigram), word-pairs (bigram) and word-triples (trigram) in a text corpus during the training phase.

n-grams having $n > 2$ can be further subdivided according to the direction in which the context is searched for neighbouring words. In a forward n-gram model, $n - 1$ preceding words are considered while in a backward n-gram model takes the $n - 1$ succeeding into account. Usually forward n-grams are used for language modelling but backward n-grams can yield better results for head-final languages like Korean and Bangla [Khan06]. A disadvantage of n-grams is that they only capture relationships between adjacent words.

2.2.4 Smoothing

The ML is a poor estimate when the amount of training data small compared to the size of the model which is generally the case in language modeling [Chen99]. The n-grams that do not occur in the training data will be assigned a zero probability. A solution to this sparse data problem is to smooth the relative frequencies to avoid zero probabilities. A simple but poor smoothing algorithm is add-one smoothing. Add-one smoothing adds one to all counts before they are normalized to probabilities. Discounting is lowering some non-zero counts to move the probability mass to the zero counts. In Written-Bell discounting, the probability mass of zero n-grams is estimated by the number of times n-grams were seen for the first time. Good-Turing smoothing is a more complicated discounting technique in which the probability mass of zero or low counts are re-estimated by the number of n-grams with higher counts.

2.3 Decoding

Decoding is the process of converting spectral measurements at the lowest level into sentences at the highest level. For each possible sentence, the probability that it produced the observation sequence is computed. In the resulting word graph, the most likely path is selected as the recognized sentence. Decoding involves the combination of various probabilistic estimators. Probabilities are assigned to these states as well as to transitions between states to determine which word is most likely to produce a particular phoneme sequence (section 2.2.2). Finally the most probable word sequence is estimated in order to (re)construct a complete sentence (section 2.2.3).

Given the observation sequence O , the most likely underlying word sequence W^* is determined by

$$W^* = \operatorname{argmax} P(W|O). \quad (2.1)$$

In the presentation above, the recognition problem is represented as the maximization of the posterior probability $P(W|O)$. Using Bayes, the posterior probability can be broken down as:

$$W^* = \operatorname{argmax} \frac{P(O|W)P(W)}{P(O)}. \quad (2.2)$$

$P(O)$ is a normalizing constant that is independent of W :

$$W^* = \operatorname{argmax} P(O|W)P(W). \quad (2.3)$$

The factor to be maximized is split into two separate probability distributions which can be modeled and trained independently of each other:

- $P(W)$ the prior probability distribution estimated by the SLM
- $P(O|W)$ the class conditional probability distribution or observation likelihood estimated by the AM

2.4 Speech Application Architectural Design

The development of speech applications is usually very complex as it involves a lot of data processing in dynamic environments. Speech recognizers from different manufacturers differ in the layout of recognition output, supported audio formats, phonetic alphabet being used, etc. and therefore require a different integration in the speech application. For every software system it holds that a good architectural design provides an abstract system view that allows designers and developers to reason about system integrity constraints and overall system behavior [White96].

2.4.1 System Overview

The diagram displayed in figure 2.8 applies to any system that performs AI-techniques on spoken content.

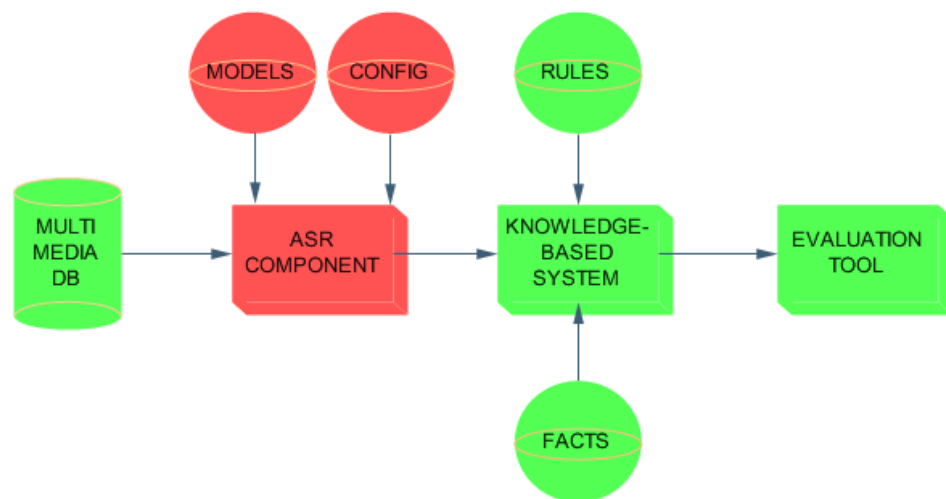


Figure 2.8: *Global design of a speech application*

The diagram shows that an ASR component pulls a multi media stream from a multi media source. The ASR component represents a non-monolithic subsystem containing the speech recognizer as delivered by a manufacturer as well as all functionality that allow the integration into a speech application. The multi media source represents any data source like a database, file-server, etc. that can provide an audio/video flow e.g. news bulletin, You Tube movie, etc.

Since there is no control process between the multi media database and the ASR component, the latter does not have any *a priori* knowledge about the nature of the multi media stream that it must process. The ASR component also pulls the three speech models described in section 2.2. Finally, a configuration file (config) is being pulled that prescribes speech engine parameters like feature extraction and search space options.

After a (successful) speech recognition, the results are pushed to a Knowledge-Based System (KBS) which is responsible for the main service of the speech application. In knowledge-based system, the time coded text is combined with other facts which results in a specific action driven by production rules [Negnevitsky02]. The evaluation tool presents any component that reports about the quality of the system by reading the output of the knowledge base.

Note that the ASR component does not receive any feedback from the knowledge-based system nor from the evaluation tool. The underlying idea is that it should not matter how the recognition results are further processed or what the nature of the KBS is. Consequently, any component succeeding the ASR component is independent of the speech recognizer being used which implies that the time coded text must always be represented in a predefined layout.

2.4.2 Pipes and Filters Architecture

The pipes and filters architectural design can be described by the analogy of a public water system. Water flows through a water pipe to a filter that adds something to or take something away from the water. Filters can be easily changed without upsetting the overall system as long as the input and output can still be streamed through the pipes. The water pipes correspond to data pipes and filters correspond to data transformations [Gamma95].

In the pipes and filters architecture, the system task is divided into sequential processing steps each separately implemented as a filter. An example of a pipes and filters architecture is the uniform pipe mechanism in Linux operating systems that provide a lot of tools that can be combined in any way with a simple shell command. Filters are connected by data pipes to form a processing pipeline. Each filter consumes data at its input and provides the transformed data at its output. "Active filters" actively push or pull data through a pipeline. Active filters can run in parallel since they are able to start to work with the partial results of its predecessors instead of having to wait for their completion [Langhorst03]. Passive filters are activated by receiving pushed data, or by receiving a pull request. Pipes connect a filter (upstream process) output to the next filter (downstream process) input. Pipes do not transform data but function as uniform interconnection buffers. Special kinds of pipes are "typed pipes", which require that the data passed between two filters have a well-defined type. The input to a processing pipeline is provided by a data source such as a text file while the output flows into a data sink such as a file or terminal. A pipeline can be subdivided into filter chains, each responsible for a specific task.

The pipes and filters architectural style allows designers to understand the overall input/output behavior of a system as a composition of the individual filters. Filters can easily be added or updated and reuse is encouraged. Concurrent execution is possible by running filters on a multiprocessor system. The pipes and filters architectural style thus supports understandability, maintainability, concurrency and re-usability. A disadvantage is that this architectural style is not really suitable for interactive systems.

3.1 Sentence Alignment in Machine Translation

Machine Translation (MT) is concerned with the automatic translation from one natural language to another. Like ASR, MT requires a lot of training data. The training data comprises texts written in multiple languages and the semantic correspondence relations between the parallel texts. In [Antonsen08], two translations of the New Testament, one written in North Sámi (source language) and the other in South Sámi, are used as training data. The semantic correspondence relations are found by applying a technique called "sentence alignment". Alternatively, subtitles can be used as parallel texts as they are freely available on the web¹ for a huge collection of movies in many different languages.

Sentence alignment is a monotonic mapping between sentences in a Source Language (SL) and sentences in a Target Language (TL). Monotonic means that crossing links are not allowed while sentences in the SL may have a different word order than words in the TL. Because the alignment is done at the sentence level, both texts need to be segmented at sentences boundaries [Tiedemann07].

A length-based approach relies on the assumption that translations tend to be of similar character lengths. Like ASR, sentence alignment introduce insertions and deletions which cause follow-up errors. Tiedemann proposed a solution called *time-slot alignment* that exploits the time information contained in the subtitles. time-slot alignment assumes that corresponding captions are displayed at approximately the same time. The time-slots of the captions in a source subtitle are mapped to a target subtitle-file (in a different language) by a set of fix-points. Time-slot alignment is a promising technique that is robust to deletions and insertions. However, the on-line available subtitles are aligned to the corresponding movie in the original language but suffer from synchronization differences because they are often 'ripped' from a DVD. In [Tiedemann08] a technique for synchronizing subtitles is proposed. This method uses *anchor points* i.e. pairs of subtitle frames which truly correspond to each other e.g. word pairs from bilingual dictionaries.

¹<http://www.opensubtitles.org>

3.2 Stream Alignment in Video Indexing

Video indexing is concerned with attaching content based labels to video for providing content based access. Although video and audio content are both valuable for video indexing, mainly video features are being used. Speech transcriptions are important features since they carry most of the semantic information. For almost all television programs, speech transcripts are available as closed captions. Closed captions also contain extra information like special markers and punctuation which could be useful as well. In order to use the closed captions for video indexing, they need to be synchronized with the video stream [Huang03].

The solution to the synchronization problem, as proposed by Huang, is a method called "stream alignment". First, the speech contained in the target video is being recognized which results in a transcription with a time code for each word (ASR stream). Next, the original closed captions (CC stream) are aligned with the transcriptions. Finally, each word in the CC stream is assigned the time code from the corresponding words in the ASR stream. If the alignment is robust enough, a word in CC might be correctly assigned a time code even if its word transcription is erroneously recognized.

The alignment algorithm of Huang is based on a dynamic programming technique for connected word recognition described by Nye [Nye84]. The Huang algorithm is a best path finding procedure that iterates word-by-word along the ASR and CC stream, until both streams have reached the last word. The CC stream is complemented with blanks to simulate insertions in the ASR stream. For each word in the ASR stream, the best matching word from the CC stream is determined only by the possible paths to it. A path in the CC stream is a match, substitution, insertion, or deletion from previous words.

The alignment score is computed by $D(i, j)$; the accumulated distance from the first ASR and CC words to the i th ASR word and the j th CC word. Table 3.1 shows the computation of $D(i, j)$ based on word identity yielding binary distances. The alignment consists of the sequence of word pairs (i, j) . Table 3.2 shows a fragment of the caption stream (CC) and the corresponding speech recognition result (ASR). One can see that due to the inserted word sequence "is done during an", the CC word "hours" misses the match with its ASR counterpart. The CC' stream represents a modified CC stream with blanks between the words "burning" and "hours" to compensate for the insertions introduced by the speech recognizer. The lower distance of CC' indicate the minimum distance approach will indeed leads to a better alignment of the caption.

Table 3.1: Computation of $D(i, j)$

Path (State)	Match (Action)	Distance (Transition cost)
insertion	ASR word i forms a word pair with a blank in the CC stream	$D(i - 1, j) + 1$
deletion	CC word j does not form a word pair with ASR word i	$D(i, j - 1) + 1$
substitution	ASR word i forms a word pair with CC word $j, i \neq j$	$D(i - 1, j - 1) + 1$
match	ASR words i forms a word pair with CC word $j, i = j$	$D(i - 1, j - 1) + 0$

Table 3.2: CC stream and its minimum distance variant

ASR	the	fires	are	nearby	woods	is	done	during	an	hours
CC	the	fire	was	still	burning	hours	after	it	started	<i>null</i>
<i>type</i>	<i>match</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>ins</i>
$D(i, j)$	0	1	2	3	4	5	6	7	8	9
ASR	the	fires	are	nearby	woods	is	done	during	an	hours
CC'	the	fire	was	still	burning	<i>null</i>	<i>null</i>	<i>null</i>	<i>null</i>	hours
<i>type</i>	<i>match</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>sub</i>	<i>ins</i>	<i>ins</i>	<i>ins</i>	<i>ins</i>	<i>match</i>
$D(i, j)$	0	1	2	3	4	5	6	7	8	8

Often, there are too much speech recognition errors for the word identity comparison method to be accurate. Therefore, Huang proposes the following improvements:

1. partial matching by word edit distance
2. comparison in phoneme space
3. partial matching by prefix comparison

In the partial matching of improvement 1, computing the word edit distance results in a match score from 0 to 1. Unfortunately, this method is too computationally expensive. Improvement 2 complicated and needs word to phoneme transformations. In the prefix comparison of improvement 3, a word pair with identical prefixes is assigned a constant score.

Part II

Analysis and Design

4 DATA EXAMINATION

In this chapter, examples of television programs, provided by the NPO, are analyzed. The programs differ in genre and potential difficulties for automatic alignment. The educational program of section 4.2 contains music, the improvisational theater program in section 4.3 is recorded in front of audience. Open captions can be included like in the news documentary of section 4.4. Closed captions may include lyrics like in the comedy of 4.5. An overview of the examined broadcasts and their difficulties is presented in section 4.6. The video frames displayed in this chapter were extracted with FFmpeg [FFmpeg]. All automatically generated speech transcriptions were obtained with the SONIC speech recognizer which is discussed in section 8.3. The observations can be found in section 4.6.

4.1 Boeken (VPRO)

"Boeken" is talk program with an interviewer and guest speakers discussing about books. Figure 4.1 shows a screen shot of Wim Brands (interviewer) and Jaap Jacobs (historian) discussing the biography about Petrus Stuyvesant in the broadcast of September 13, 2009. The speech is clear and rather formal. However, crosstalk, the use of English words, dissimilarities in word order between the captions and can be difficult for automatic alignment. Crosstalk heavily frustrates speech recognition and results also in overlapping captions which can not be rendered as such.



Figure 4.1: Wim Brands (left) and Jaap Jacobs in "Boeken"

23. Die wordt groot gevierd in New York. We zitten midden in die viering.
24. Er zijn veel misverstanden. Er is veel ruis en onzin.
25. Ik las net al: Jan Kees, Yankees. Breukelen, Brooklyn.
26. Broadway, Bredeweg. Dat weten we allemaal.
27. Maar dit. Jan Kees werd Yankees. Klopt dat?
28. Eh, ik ben benieuwd.
29. Ik heb dit boek nog niet in detail bekeken.
30. Ik ben benieuwd wat hij erover te zeggen heeft.
31. Ik heb zo'n verhaal eerder al wel gelezen.
32. Maar het lijkt mij meer een oppervlakkige klankovereenkomst...
33. dan een echt etymologische oorsprong.

Figure 4.2: *Captions from "Boeken"*

One can see in textbox displayed figure 4.2 that captions 23, 25, 26 and 27 contain the names "New York", "Yankees", "Brooklyn" and "Broadway" respectively. As these names are English, they will be hard for the Dutch trained speech recognizer. This can be easily seen by inspecting phonetic transcription used by the SONIC speech recognizer. The word "Yankees" will be incorrectly pronounced in Dutch as "iejankees" and the word "new" will not be pronounced as "nieuw" but literally as "new" which is not a Dutch word. The only exception is "York" which will be pronounced as "jork" by most Dutch people. An example of a complete SONIC lexicon can be found in appendix B.

4.2 Iedereen kan aquarelschilderen (Omroep MAX)

"Iedereen kan aquarelschilderen" is an educational television program about how to make aquarel paintings. There is only one speaker, paintress Inge Schagen. Although she is probably reading text from auto-cue, there are some small differences between the captions and the actual speech. The subjects are separated by 5 seconds of music. Figure 4.3 shows a video frame of the first lesson, the broadcast of August 8, 2008, in which the necessary materials are discussed. The captions are displayed in the textbox in figure 4.4. The captions look much like manual instructions.



Figure 4.3: Video frame of animation synchronous to the music fragment

17. De afstand met een gewone tafel is prima.
 18. Dan heb je een goede afstand en kun je goed zien wat je doet.
 19. Mensen die moe worden in hun benen kunnen een hoge kruk pakken.
 20. Zorg dan wel dat je flink hoog boven de tafel blijft zitten.
 21. Wat is er nodig voor het maken van een aquarel?
 22. Allereerst water.
 23. Neem gewoon een lekkere oude pot.
 24. Water uit de kraan is prima om mee te werken.
 25. Vervolgens verf, aquarelverf wordt verkocht in tubes.
 26. Of in napjes.

Figure 4.4: Captions from "Iedereen kan aquarelschilderen"

Figure 4.5 displays a part of the audio signal where speech is alternating with music. The blue segment indicates a silence, the red segment indicates music (lack of silences), the other parts are normal speech. Silences are easy to locate since they have a zero amplitude while music seems to have no silences at all.

Consider the recognition result: *..hoog boven de tafel blijft zitten **voor een** wat deze nodig voor het maken van aquarellen..*

The bold words are insertions caused by the music fragment that is located between caption 20 and 21. A tool for automatic music detection [Seyerlehner07] would be very useful in this situation as it can tell the aligner which part of the speech transcription must be ignored. The music pauses indicate a change of the subject which is also graphically signaled by short animation synchronous to the music fragment (see figure 4.3). So video boundary detection [Wang02] could be helpful as-well.

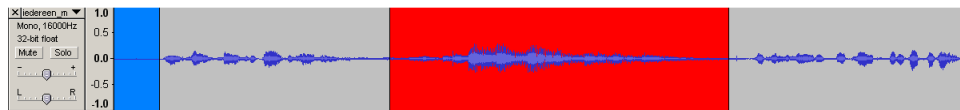


Figure 4.5: Speech signal containing silence (blue) and music (red)

4.3 De vloer op (HOS)

"De vloer op" is an improvisation theater program, written by Peter de Baan. Improvisational theater is a form of theater in which the actors, given a global scenario description, perform spontaneously. The performances are recorded in the Desmet Studio's in front of a live audience. Since this kind of theater tends to be comedic, there is often hearable laughter.



Figure 4.6: Video frame of "De vloer op"

Figure 4.6 displays a video frame from the episode "Mijn leven als moslima" from the broadcast of June 27, 2008. Saskia Temmink plays a Muslima that wrote a confronting book about the Islamic society, based on her personal experiences as the wife of an Iranian politician. Pierre Bokma plays the publisher who is terrified to publish it.

19. Ik weet niet hoe lang dat gaat duren. Dat weet ik niet. 20. Ik bel je als het zover is. Hoi. 21. Bij Tantulus. Er was nog een plekje. Leuk, he? 22. Oke, lieverd. Ik bel je. Ja, daag. Daag. 23. Dag. 24. Zo... Ha ha, je knoop. Och, kind! Ooo... 25. Van de vorige sessie! 26. Hier, kijk eens. Prachtig. Ja, net als dit. 27. Ja? Ja. Ja, ik dacht... mwah. 28. Zeg, wat een mooi boek! Ik ben zo nerveus. Echt waar? 29. Jaaaa. Wacht even.

Figure 4.7: Captions from "De vloer op"

The text-box in figure 4.7 shows the last ten captions. The captions indicate the informal nature of the dialog by verbal expressions like "Ooo" and "mwah". Note also that the captions sometimes lack a separating space which could be a problem for automatic alignment too which may lead to an improper word segmentation.

4.4 NOVA (NPS)

NOVA is a news documentary program which provides background information about news items. The speech in NOVA usually comprise news reading, interviews and discussions in rather formal Dutch. Figure 4.8 displays a video frame of the NOVA broadcast January 22, 2009 containing a documentary about primatologist Marc van Roosmalen who was arrested in Brazil. Television presenter Twan Huys is a very fast speaker while Marc van Roosmalen speaks rather slow. The differences in speaking rate could have a negative effect on an automatic aligner if it expects a constant word rate.



Figure 4.8: Twan Huys reporting about Marc van Roosmalen

The captions were provided in two different formats: *verbatim* and *condensed*. The verbatim captions contain literal speech transcriptions which was being text reduced to condensed captions. The text-box in figure 4.9 shows a part of the verbatim captions and their corresponding title numbers. The title numbers of condensed captions that are different from the corresponding verbatim version are indicated by a postfixed quote. The total number of words of the verbatim captions amounts 872 while there are only 755 words in the condensed format resulting in a text reduction of almost 87%. Text reduction may frustrate automatic alignment as it results in a mismatch between the lengths of the actual speech transcription and the corresponding caption.

58. De pers is van tevoren ingelicht.
 59. De apen en de kooien worden in beslag genomen.
 59'. De apen en kooien worden in beslag genomen.
 60. NOVA zoekt Van Roosmalen op in 27,als hij even vrij is...
 61. in afwachting van het oordeel van de rechtbank op de eis van 14 jaar celstraf.
 62. Is dat een normale straf voor een vergrijp als dat?Nee.
 62'. Is dat een normale straf dat?Nee.
 63. Je kan gewoon zien dat het een politieke,niet een tintje, maar een politieke tintheeft.
 63'. Daaraan zie je dat het een politieke tintheeft.
 64. Dat ze mij... Het is een excuus?
 65. Ja,een excuus omdat ik gewoon, terwijl ik mijn onderzoek deed...
 65'. Ja,omdat ik onderzoek deed
 66. Ik ben heel Amazonia doorgevaren met mijn onderzoeksboot.
 66'. Ik ben heel Amazonia doorgevaren
 67. Ik heb gewoon zonder te willen als wetenschapper te veel gezien.
 67'. Ik heb daardoor zonder te willente veel gezien.
 68. MUZIEK

Figure 4.9: Captions from NOVA

The waveform of figure 4.10 shows background music in the red part that merges into a new speech fragment in the green part. Music detection alone is not sufficient in this case; since the speech and music overlap, the music must be suppressed to enhance the intelligibility. The music part that does not overlap with the speech is indicated by the word "MUZIEK" (caption 68).

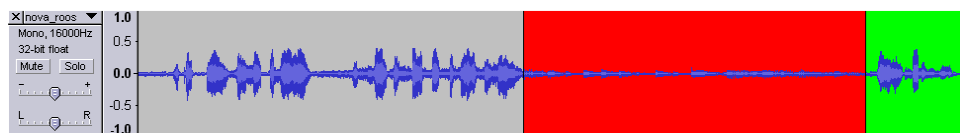


Figure 4.10: Background music with (green) and without speech (red)

Since the recording contains a 16 seconds during speech fragment in the Brazilian language, some of the video-frames have open captions attached to it e.g. the video frame in figure 4.11. The Brazilian speech is located between closed captions 58 and 59. Since the open captions are directly attached to the video, they are not reflected in the closed captions. As the speech recognizer was trained only for the Dutch language, it produces a meaningless string of 34 words. These insertions could be a major problem for automatic alignment.



Figure 4.11: Screenshot of subtitled Brazilian speech

The end of this broadcast contains a political cartoon animation¹ in which the Dutch minister "van Middelkoop" falls on the ice. Figure 4.12 shows a video frame of this animation. The corresponding sound is captioned as "BONS" which is Dutch for "BOOM". It will be problematic to correctly align non-speech captions automatically without any means for automatic speech/non-speech detection e.g. the method described by Mangti et. al [Maganti07].



Figure 4.12: Cartoon requiring captions of non-verbal sounds

¹made by Joep Bertrams

4.5 't Vrije schaep (KRO)

A seventies retro (see figure 4.13) comedy series written by Frank Houtappels. Some speakers have strong regional (Amsterdam) accents. Figure 4.13 shows a video frame of the episode "Jacqueline", the broadcast of February 2, 2009.



Figure 4.13: *MarcMarie Huijbregts and Loes Luca in 't Vrije schaep*

The sound track contains background sounds like rain and cafe hubbub. Some of the speech merge into songs. Figure 4.14 shows a part of the audio signal where a song is slowly started from a dialog. The yellow segments highlight piano music while the green segments indicate a combination of the piano music with speech.

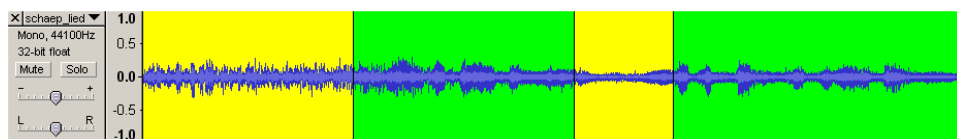


Figure 4.14: *Piano sound with (green) and without (yellow) speech*

The text-box in figure 4.15 shows a part of the captions. Captions 46, 47 and 48 are a spoken answer to the question of caption 45 which merges into a song starting at caption 49. The song must thus be subtitled as well. Note that the lyric captions are prefixed by a star. Assuming a version of the system that is capable of detecting music, the captions can tell whether or not the music has to be ignored.

45. Heb-ie een ander? Niet dat ik weet.
 46. Dat je dat niet weet, opoe. En datje dat nooit aan ons heb verteld!
 47. Wat zal ik zeggen? Het ging allemaal een beetje geruisloos. Geruisloos.
 48. Ja. Er was nooit geen mot of heibel of niks.
 49. *Toen hij de deur dichtdeed met z'n paspoort in z'n jas
 50. *Had zij totaal geen weet en stond de snelkookpan op 't gas
 51. *Ze zouden net gaan eten, de vier placemats lagen klaar
 52. *Er stonden oude auto's op en hij droeg half lang haar
 53. *Ze hadden het zo goed, ze hoefden niks te laten staan
 54. *Hij deed wat in computers dus dat was een prima baan

Figure 4.15: *Captions from 't Vrije schaep*

4.6 Observations

An overview of the television programs can be found in table 4.1. The speech in television programs can be formal, instructive, spontaneous or overlap e.g. in discussions. Broadcasts can contain music located at the begin or end but may interrupt or even merges the speech as-well. Language dissimilarities can occur like regional accents, loanwords or even complete sentences in a foreign language. Background noise caused by audience or as part of a scene. Table 4.2 indicates the difficulties for each example program. It is important to realize that a non-speech sounds is a different issue than background noise since the latter does not have to be subtitled. A similar difference exist between songs and background music.

The expected difficulties - for automatic alignment - that were found during the analysis of the example data can be subdivided into the following categories:

1. textual representations that mismatch the speech:
 - * captions with a different word order than the corresponding speech
 - * condensed captions (text reduction)
 - * captions with type errors like missing spaces
 - * captions for non-speech sounds like "BONS"
2. language dissimilarities:
 - * regional accents
 - * names in a different language like "New York" and "Yankees"
 - * filler words like "Ooo" and "mwah"
 - * speech in a different language e.g. Brazilian
3. speech quality issues:
 - * differences in speaking rate
 - * crosstalk
 - * background noise like rain and laughter
 - * interrupting music

Table 4.1: *Overview of analyzed data*

television	duration	number of titles
Boeken	25mn 51s	385
Iedereen kan aquarelschilderen	9mn 24s	142
De vloer op	24mn 14s	357
NOVA	7mn 54s	102
't Vrije schaep	11mn 38s	176

Table 4.2: *Overview of difficulties in the analyzed data*

television program	category 1	category 2	category 3
Boeken	different word order	English words	crosstalk & different speaking rates
Iedereen kan aquarelschilderen	different word order	none	interrupting music
De vloer op	non-speech captions	filler words	background noise
NOVA	condensed captions & non-speech captions	speech in a different language	background music & different speaking rate
't Vrije schaep	lyric captions	regional accents	song & background noise

5 REQUIREMENTS

In this chapter the required behavior of the system for automatic closed caption alignment is analyzed. Each section discusses a specific task or sub-process and what it is supposed to do. The goal of the system is to process the video- and subtitle file and return a subtitle file in which the captions are attached by correct time codes. For each subtitle the *in-time* is determined by the speech recognizer while the end-time is estimated by both the in-time and the display- time. The display-time is determined by *title heuristics*, listed in section 5.1. Therefore we need a means of data communication which is discussed in section 5.2. After receiving the files, their content must be stored in a format that is supported by the system. On the other hand, the output of the system must be compatible with the software of the user. These issues are the topic of section 5.3. Speech recognition relies on statistical models as described in section 5.4. Speech recognition involves many parameters; the topic of section 5.5. Time-codes are found by aligning the captions with the recognized speech, this is discussed in section 5.6. To detect the cause of possible low performance, errors or other unwanted effects all actions need to be logged as described in section 5.7.

5.1 Speech Estimating Title Heuristics

An interview with subtitle editors of TT888 about the required timing of captions resulted in the list of heuristics below. The combination of speech technology and heuristics motivated to name the proposed system for automatic caption alignment *SETH*¹: Speech Estimating Title Heuristics.

1. A pause between two captions has a maximum duration of one second. This is to allow smooth transitions between the captions and to avoid flicker.
2. To allow a viewer enough time to absorb a caption's content and still watch the action of the program, a maximum reading rate is applied to caption text. The default reading rate is 145 words per minute but should be adjustable to a maximum of 200. For children programs the reading rate is limited to 170.
3. The start time of the display of a caption must approximate the start time of the corresponding speech. It will be easier to separate different speakers without a different coloring of the captions.
4. The caption "TT888" does not have to be aligned but must be displayed during the first 4 seconds.
5. The in- and out-time of titles with a too low confidence.

¹This is an acronym of the Ancient Egyptian God of chaos who killed Osiris

5.2 Data Communication

Figure 5.1 shows the context diagram of the system. It represents the interactions between SETH and the user (an editor at TT888), and between SETH and an operator (an employer at Dutcheer). The user must be able to send a broadcast video file and a subtitle file containing the captions. Because the speech characteristics in broadcasts can strongly differ, the user must be allowed to change some parameters analogous to differences in manual alignment between different kinds of broadcasts. After sending the files, the user needs some kind of feedback to know that the transmission of files succeeded and that they are being processed. Finally the user must receive the new subtitle-file.

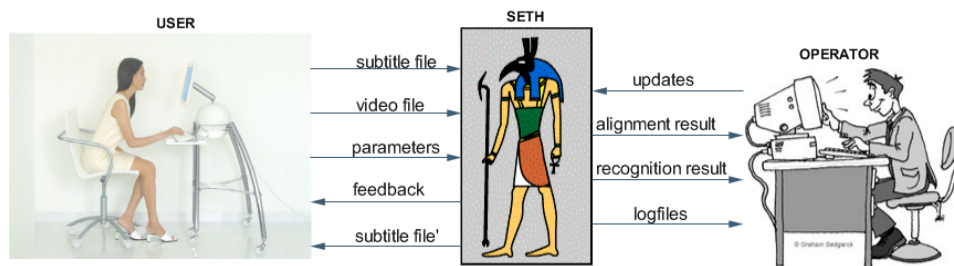


Figure 5.1: *Context of SETH*

5.3 Pre- and Post-processing

To produce an ASR stream, the audio signal must be extracted from the video file in a format that is supported by the speech recognizer. The required bit rate depends on the acoustic model that is being used during speech recognition. Models for microphone speech require 16 kHz while models for telephone speech expect 8 kHz speech. Some speech recognizers are unable to handle file headers which are included in e.g. PCM WAV files. The import process is responsible for presenting the speech in a suitable audio format.

The subtitle file that is provided to the system, contains captions and the corresponding time-codes are initialized to zero. The import process is concerned with extracting the captions from the subtitle file at the input of the system. The function of the export process is to return a subtitle file with the same captions in the same format as the input subtitle file. Note that the only difference between the import subtitle file and the export subtitle file is that is that the latter contains time codes estimated by the alignment process. Convention about subtitle file format is essential as any small deviation can lead to corrupt files. To check whether or not the export of captions is done correctly one can simply rerun the import process to convert the export subtitle file into readable text.

5.4 Model Training

From chapter 2 it is known that the ASR process requires three, language dependent, (statistical) speech models:

- * a pronunciation model or lexicon representing words by phonetic pronunciations
- * one or more language models to estimate the most likely word sequence
- * an acoustic model to estimate the most likely observation sequence given an utterance

A sufficient ASR quality requires that the models that are up to date and relevant to the application domain [Wiggers08]. Consequently, SETH must provide a means for regular model updating i.e. training on newly available training text. Generally, any statistical model used by the system must always be re-estimated on newly available data.

5.5 Speech Decoding

The decoding process i.e. conversion from speech to text is started by invoking speech recognition software provided with a list of all parameters defining the models to be used, the audio file, the sample rate, the feature extraction method, word penalties, etc. Output parameters involve what kind of output must be produced and where to store it. Since the alignment needs time information, the speech recognizer must be configured such that it will also output the time stamps. As is the case for the audio file and the statistical models, inaccuracies of the parameters cause the recognition process to fail.

5.6 Alignment

The alignment process is responsible for synchronizing the captions with the speech. From section 1.5 it is clear not to assume a perfect speech decoding and that not all captions contain literal speech transcriptions. Hence, the alignment has to deal with uncertainty. Therefore the alignment must tolerate partial syntactical mismatching. Besides tolerance, robustness is important to avoid that one misaligned caption will cause a failure of succeeding alignments (snowball effect). Depending on the broadcast, the number of transcriptions produced by the speech recognizer can be very large, therefore some kind of pruning needs to be done to prevent the alignment process from wasting time and memory or getting lost in an irrelevant part of the search space. Finally, the alignment must offer transparency i.e. it must allow the inspection of its decisions.

5.7 Logging, Error Handling and Updating

An error that occurs anywhere in the system must be caught to avoid a very poor alignment caused by e.g. a wrong filename or different file format. System monitoring must be provided to detect possible errors. Therefore all actions of SETH must be logged. Together with the speech recognition and alignment results, log-files must be available to operators so that they can analyze possible bugs or inaccurate models. Deployment of new releases by operators must be possible as-well.

6.1 System Overview

In figure 6.1 an overview of the system is presented. The inputs are a video-file and a subtitle file (without time codes). The application that is responsible for the caption alignment is called "Npobuilder". Npobuilder is a web-application that runs on a server. It produces a subtitle file with time codes using heuristics and a speech recognizer. Speech models are trained by Npobuilder and provided to the speech recognizer via the server. The system contains an evaluation tool for quality monitoring.

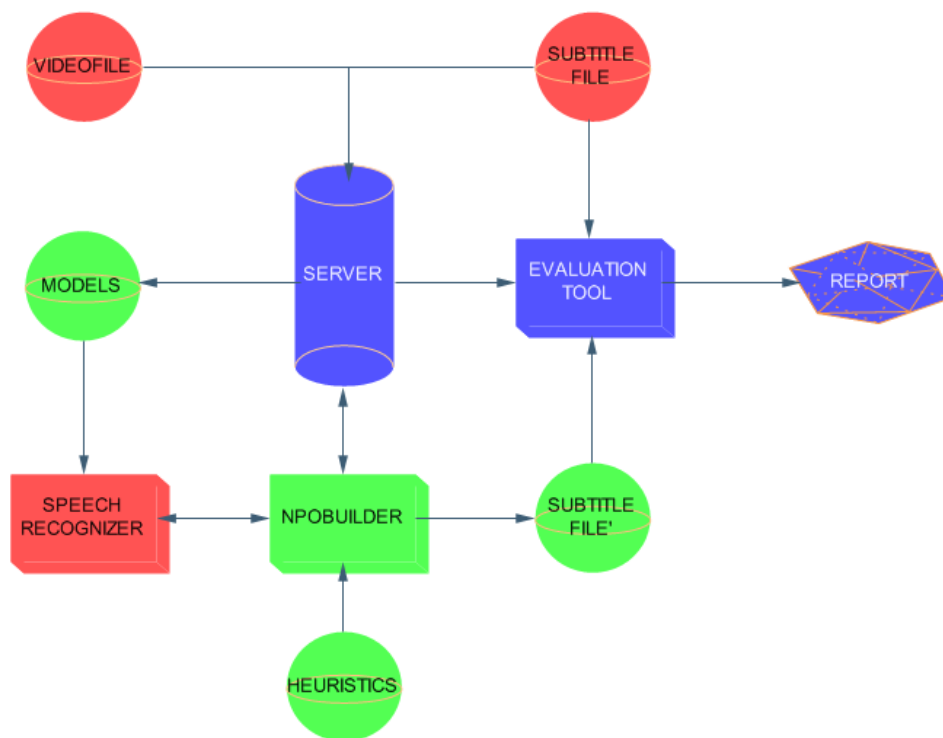


Figure 6.1: *System overview of SETH*

To clarify the concepts with respect to the input and output of SETH, figure 6.2 shows an Entity Relationship Diagram (ERD). Each broadcast has a video file and subtitle file. A subtitle file has more than one subtitles. Each subtitle can contain more than one caption. An aligned subtitle is a subtitle for which the start and end time are determined by a hypothesis i.e. a string produced by the speech recognizer which matches the caption to a certain degree (score).

6.2 System States

The high-level processes can be explained by regarding them as the different states the system can adopt as being displayed in figure 6.3. The different colors indicate the kind of actions that are performed in a particular state. In the orange states, Linux shell commands are executed¹. Blue indicates the involvement of the speech recognizer. Conversions are done during green colored states while red indicates the core alignment process.

In the idle state, the system is checking whether there is something to do. During the import state, the captions are read from the subtitle file and stored in a xml-file to increase the accessibility. The audio stream is extracted from the video file and stored in multiple audio files to allow simultaneous processing of multiple speech recognizer instances. The audio-file names are written in a list that will be consulted by the speech recognizer. In the train state, the captions are cleaned and stored in a text file that is suitable to generate a language model. Words are stored in a dictionary which is extended with corresponding phonetic descriptions to form a pronunciation model. The speech recognition process is performed during the test state, after initializing the speech recognizer with a configuration file. The alignment state involves two alignment passes. Time codes are stored in a new subtitle file during the export state. The subtitle file is made available for downloading. All received and generated files are moved to a compressed archive on the server to spare storage capacity and prepare the system for new input. This state is called the cleaning state.

¹For more information about Linux shell programming, the reader is referred to [Mitchell01].

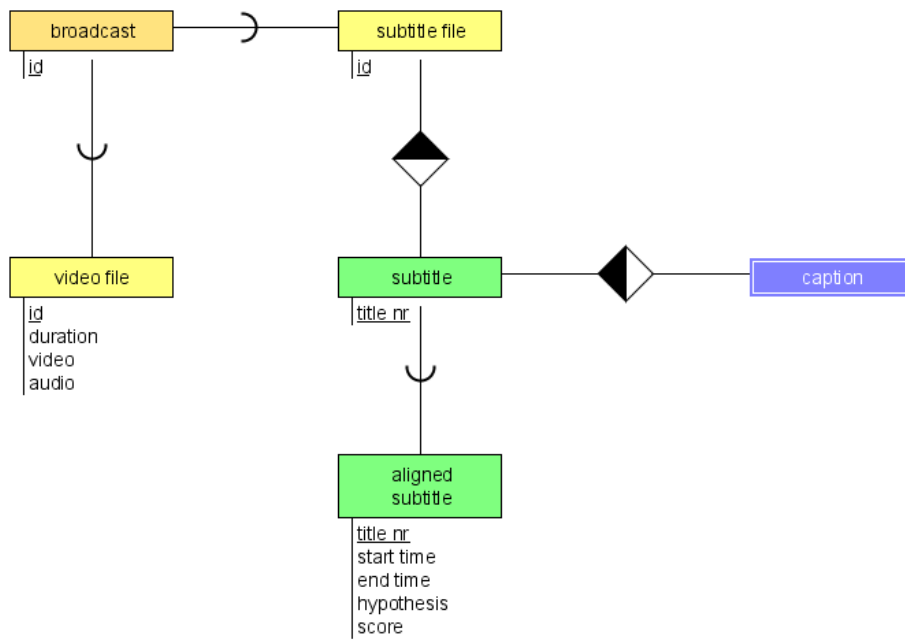


Figure 6.2: *Entities used by SETH*

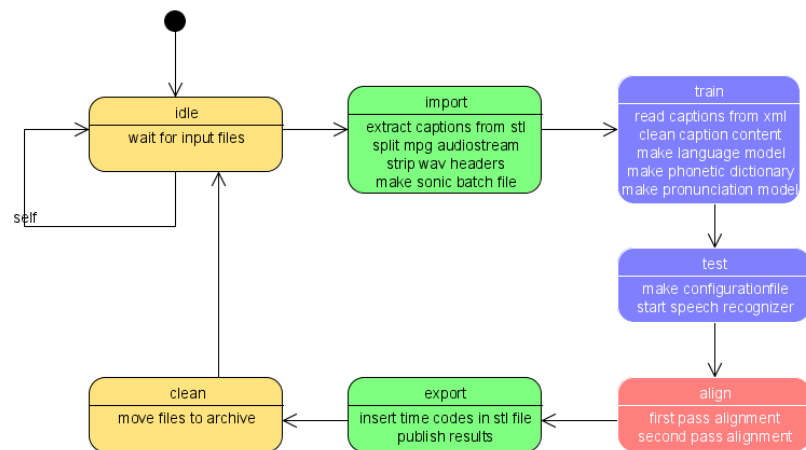


Figure 6.3: *States of SETH*

6.3 Speech Models

A sufficient ASR quality requires that the speech models are up-to-date and relevant to the application domain [Wiggers08]. Consequently, the speech application architecture must provide a means, at least manual, for regular model updating i.e. training on newly available training data. The acoustic model is the most general model as it defines how phonemes are being produced in a specific language. For every recognition the same acoustic model is used since it is uimpracticable to train a new model for every sound-track automatically. The aim of the ASR in this particular speech application is to identify the time locations of the CC words. In order to optimally reflect the captions, both the lexicon and LM should be trained on them. Since the captions differ for every broadcast, a generic architecture is needed to allow the construction of new models in a consistent manner. In the paper "Architectural Design for Speech Applications" in appendix A a generic pipes and filters architecture for speech applications is presented. The design contains a component called "speech model pipeline" (figure 6.4) which allows the construction of speech models from arbitrary text sources.

6.3.1 Speech Model Pipeline

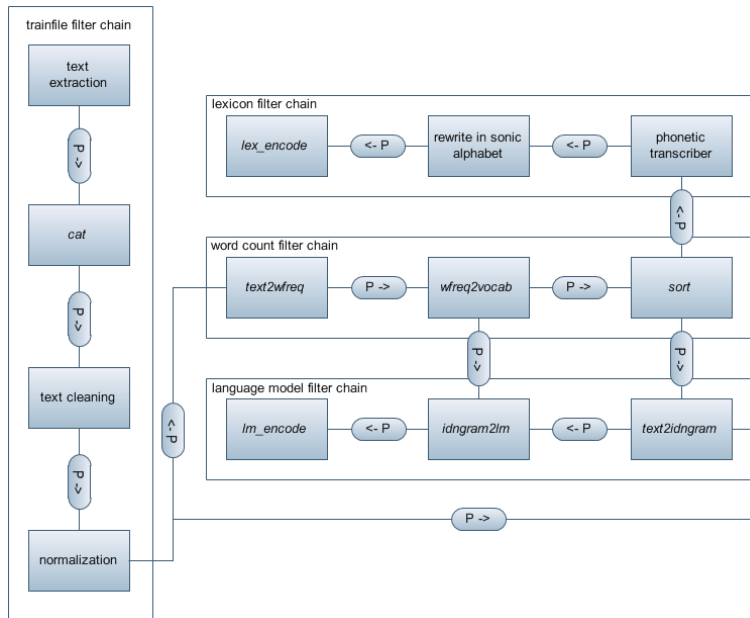


Figure 6.4: Speech model pipeline

The specification of the filter chains as components of the speech model pipeline can be found in table 6.1. The output of the speech model pipeline is binary which means that the speech models must be encoded into binary files that are compatible with the speech recognition software. As can be seen in figure 6.4, these two binary speech models are produced by two separated filter chains at the end of the pipeline. Since they do not depend on each other, the lexicon and language model can be created simultaneously. The pronunciation and language model filter chain share the word count filter chain.

Table 6.1: *Specification speech model pipeline*

filter chain	function	input type	output type
train-file filter chain	create train-file	document	ascii
word count filter chain	make statistics	ascii	ascii
lexicon filter chain	make lexicon	ascii	bin
language model filter chain	make language model	ascii	bin

The train-file filter chain comprises a number of sequential text transformations to represent the lingual context of the application in a train-file. The support of different document types implies text layout conversions including OCR . All text resources are collected by the Linux filter *cat* (see table 6.2). Text cleaning involves the conversion of upper-case to lower-case characters and the removal of punctuation marks and unwanted white-spaces. Text cleaning is necessary to avoid that different occurrences of the same word are regarded as different words [Klerk06]. The text cleaning filter also replaces digits by words. The normalization filter adds structure to the text like sentence and paragraph tags.

Table 6.2: *Specification train-file filter chain*

filter	function	input type	output type
text extraction	extract text from documents	document	ascii
cat	compose text stream	ascii	ascii
text cleaning	make proper layout	ascii	ascii
normalization	add structure	ascii	ascii

Table 6.3: *Specification word count filter chain*

filter	function	input type	output type
text2wfreq	determine word frequencies	ascii	wfreq
wfreq2vocab	make vocabulary	wfreq	vocab
sort	sort vocabulary	vocab	vocab

Table 6.4: *Specification lexicon filter chain*

filter	function	input type	output type
phonetic transcriber	make phonetic transcriptions	vocab	lex
rewrite in sonic alphabet	convert phonetic transcriptions	lex	lex
lex_encode	make binary lexicon	lex	bin

Table 6.5: *Specification language model filter chain*

filter	function	input type	output type
text2idngram	make n-grams	ascii & vocab	idngram
idngram2lm	make language model	vocab & idngram	arpa
lm_encode	make binary language model	arpa	bin

Table 6.6: *Origin off-the-shelf filters*

filter	origin
ffmpeg	web
sox	web
sonic_batch	sonic
cat	linux command
text2wfreq	cmu-cam toolkit
wfreq2vocab	cmu-cam toolkit
sort	linux command
lex_encode	sonic
text2idngram	cmu-cam toolkit
idngram2lm	cmu-cam toolkit
lm_encode	sonic

The word count filter chain implements filters from the CMU/Cambridge Statistical Language Modeling Toolkit [CMU-CAM] that is described in section 7.2. The specification in table 6.3 shows that filter *wfreq2vocab* transforms word frequencies to a vocabulary. As can be seen in the diagram in figure 6.4, the vocabulary can be reused in the lexicon filter chain.

The idea behind the lexicon filter chain, specified in table 6.4, is to provide a means for automatic lexicon creation in different phonetic notations. The design allows changes in the phonetic representation to be performed in a conversion filter (rewrite in sonic alphabet) leaving the phonetic transcriber unchanged. It is the responsibility of the conversion that the lexicon satisfies the layout as being specified by the speech recognizer tool that integrates the lexicon (lex_encode in SONIC).

The language model filter chain only contains off-the-shelf filters which means that only the pipes have to be developed. Table 6.6 shows the off-the-shelf filters being used in this design and their origin. The key concept is that the design indicates which building blocks need to be developed and how they can be combined with existing tools.

6.4 Alignment Algorithm

Both Robert-Ribes [Robert-Ribes98] and Jongebloed [Jongebloed08] state impossible to find the exact words of closed captions in the audio signal. Therefore, all speech must be recognized before it is aligned with the captions. This suggests a stream alignment approach like the dynamic programming approach developed by Huang (section 3.2). Although the Huang approach has an elaborate compensation for speech recognition errors, it is very computationally extensive and lacks a proper word edit distance. Therefore, in section 6.4.2 a Huang like approach is discussed that is only applied fragments rather than the complete streams.

In the sentence alignment algorithm developed by Tiedemann (section 3.1), time-slot information is being matched to identify corresponding sentences in parallel texts. Caption alignment is the opposite problem; find the time-slots of the CC sentences by matching them with ASR sentences. Regarding the CC stream as source language and the ASR stream as target language, sentence alignment can be applied by implicitly segmenting the ASR stream at the caption boundaries. A length-based approach can be used since the length of a caption is expected to approach the length the corresponding utterance. The length-based CC alignment algorithm is discussed in section 6.4.1. The proposed alignment algorithm comprises a rough length-based approach during the first pass followed by a fine tuning stream alignment during the second pass.

6.4.1 First-Pass Alignment

Since the audio stream will also contain filler words, background speech, music, etc. and the captions could be text reduced, the sentence boundaries of the ASR stream will not match the caption lengths exactly. Hence, some kind of word identity comparison is necessary for synchronization.

Because the captions share a significant number of words, not all words are useful as anchor points. In [Ramos03] the metric Term Frequency Inverse Document Frequency (TF-IDF) is examined to determine what words in a corpus of documents might be more favorable to use in a query. TF-IDF is an estimate for relevance of a word in a particular document. It is computed by the relative frequency of words compared to the inverse proportion of that word over the entire document corpus.

In table 6.7 a part of the word counts of the captions of television program "Iedereen kan aquarelschilderen" from section 4.2 can be found. The table shows commonly frequently occurring words like "een", "en" and "er". Although the word "aquarel" has a TF of 8, the words that stem from it like "aquarelblok" en "aquarelpapier" occur only once. The complete list of word frequencies can be found in appendix C.

Table 6.7: Word counts of "Iedereen kan aquarelschilderen"

word	count
ansichtkaartje	1
aquarel	8
aquarelblok	1
aquarellen	1
aquarelletje	1
aquarelpapier	1
aquarelschilder	1
atelier	1
een	50
eens	4
en	34
er	20
is	30

Term frequencies can thus be estimated by concatenating all captions and generating a word frequency list. Regarding each caption as a document, the inverse document frequency can be estimated by the total number of sentences minus the number of sentences that do not contain the term. Assuming that there are no repetitions of a word one sentence, the number of sentences that do not contain the term can be estimated by the term frequency. Hence the $TF - IDF$ of a word w can be computed by

$$TF - IDF(w) = \frac{\#w}{\#sentences - \#w}. \quad (6.1)$$

Since there are no real documents, the TF-IDF will be large for frequently occurring words and small for words that appear rarely in the captions. By taking the logarithm, the low frequent words will get a larger weight:

$$weight(w_i) = -^2\log(TF - IDF), weight(w_i) = -^2\log\left(\frac{\#w}{\#sentences - \#w}\right). \quad (6.2)$$

Note that this weight measure shows a similarity with the Shannon information measure which states that low probable events carry more information than high probable events [Boekee91]. One can think of the length-based approach as sliding a window word-by-word over the ASR stream and selecting the best matching window position. The best position is determined by anchor points. An anchor point is any CC word that equals the ASR word at the same position. Each window position is a candidate sentence pair which score is computed by:

$$score = \sum_{i=0}^n weight(w_i). \quad (6.3)$$

For example, consider the alignment of the caption

"Dit is de eerste les van de cursus: Iedereen kan aquarellen." This caption can be broken down into CC stream:

["dit", "is", "de", "eerste", "les", "van", "de", "cursus", "iedereen", "kan", "aquarellen"]

Since this CC stream contains 11 words, a window of size 11 is shifted over the ASR stream:
["een", "is", "eerste", "les", "van", "het", "is", "is", "iedereen", "kan", "aquarellen"]

The first matching word is "is" and the last matching word is "aquarellen". The weights of these words are calculated:

$$weight('is') = -^2\log(\frac{30}{112}) = 1.90 \text{ bit},$$

$$weight('aquarellen') = -^2\log(\frac{1}{141}) = 7.14 \text{ bit}.$$

Clearly, the word "aquarellen" is assigned a larger weight than "is" reflecting the property of being stronger anchor point. Since text reduction only eliminates less important words [Ahmer02], this anchoring method will be more robust to text reduction than a pure word identity comparison method. Summing the weights of the four matching words gives a score of 21.7 bit. A less accurate candidate is retrieved by shifting the window one word to the left.

ASR : *["een", "een", "is", "eerste", "les", "van", "het", "is", "is", "iedereen", "kan"]*
 CC : *["dit", "is", "de", "eerste", "les", "van", "de", "cursus", "iedereen", "kan", "aquarellen"]*

Although the words "eerste" and "les" both have a weight of 7.14 bit, the total score of the three matching words is 17.3 bit which is near but below the winning the score of 21.7 by the winning candidate. A list of all non-zero score candidates for this caption can be found in appendix D.

As CC stream comprises the first caption, one would expect a match at the 0th position instead of the position two words shifts away from the start. This can be explained by the introduction music which causes insertions into the ASR stream.

Figure 6.5 shows a flowchart of the first-pass alignment algorithm. The component "Match" is responsible for the sliding window algorithm described above. The variable offset is the out-frame number of the last matching word of the winning candidate and is used as a kind of sentence boundary indicating the 0th window position of the next caption. If the score of the winning candidate is larger than a predefined threshold, the start time of the caption is affiliated with the in-time of the first word of the candidate. The out-time of the caption is calculated by

$$out - time = in - time + disptime, \text{ where}$$

$$disptime = \max(\frac{\#words*60}{readingrate}, 1).$$

6.4.2 Second-Pass Alignment

Captions for which no sufficiently matching ASR sentence are found in the First-Pass Alignment (FPA), are selected for a Second-Pass Alignment (SPA). The number of ASR words that are not associated with a time-slot is a monotonic non-increasing function, so after the FPA, the search space is probably reduced. This implies that unaligned captions can be globally positioned in the time-gaps of their aligned neighbors. So if e.g. captions 13 and 15 have been aligned successfully, the time-slot of caption 14 can be linearly interpolated by the out-time of caption 13 and the in-time of caption 15.

More generally:

$$out - time(i - 1) \geq in - time(i) \text{ and } out - time(i) = in - time(i + 1)$$

where i is the title number.

The result of this linear interpolation is an association between a caption and a fragment of the ASR stream. Except from substitutions, the mismatch during the FPA can be caused either by insertions or deletions in the ASR stream. In case of insertions, the #words in the ASR fragment > #words in the caption. In contrast, deletions result in an ASR stream that is too short to be matched with the caption. Like in the Huang algorithm, the insertions are compensated by blanks in the CC stream.

Table 6.8: *Alternative caption simulating insertions*

ASR	is	dat	een	normale	straf	voor	een	vergrijp	als	dat	nee
CC	is	dat	een	normale	straf	nee					
CC'	is	dat	een	normale	straf	INS	INS	INS	INS	INS	nee

For each deletion, a word in the CC stream will be deleted. Insertions can be introduced by the speech recognizer e.g. as a result of background music but may arise from text reduction. Consider e.g. caption 62 and its condensed counterpart caption 62' in figure 4.9. The caption is reduced by leaving out the sub-phrase "voor een vergrijp als dat". Regarding the verbatim caption as ASR fragment, an ASR-CC match is possible by inserting 5 blanks in the condensed caption as shown in table 6.8.

For each caption that has been selected for the second pass, one or more alternative captions are generated which simulate insertions or deletions at different word positions. Figure 6.6 shows an extended version of the ERD with the new entity *alternative caption*.

The simulation of insertions or deletions will always result in two equally sized streams allowing the use of the sliding window algorithm from the FPA. The number of window shifts n will be lower than in FPA as the ASR search space is reduced. But instead of one window, there now are m windows where $m = \# \text{alternative captions}$. This will yield $m * n$ scoring calculations which will strongly raising the computation time for large sentences. Therefore, the number of alternative captions is reduced by a distance measure. The Levenshtein Distance (LD) is a string similarity measure. The distance is determined by the number of character deletions, insertions, or substitutions required to transform a source string into a target string. The lower the Levenshtein distance, the more equal the strings are ². Table 6.9 shows three alternative captions and their Levenshtein distance with respect to the ASR stream fragment. Figure 6.7 presents a flow chart of the complete SPA process.

Table 6.9: *Alternative captions and LD*

ASR	aquarellen	allereerst	een	hele	water	
CC	allereerst	water				
CC1	INS	INS	INS	allereerst	water	20
CC2	INS	INS	allereerst	INS	water	18
CC3	INS	allereerst	INS	INS	water	17

²<http://www.merriampark.com/ld.htm>

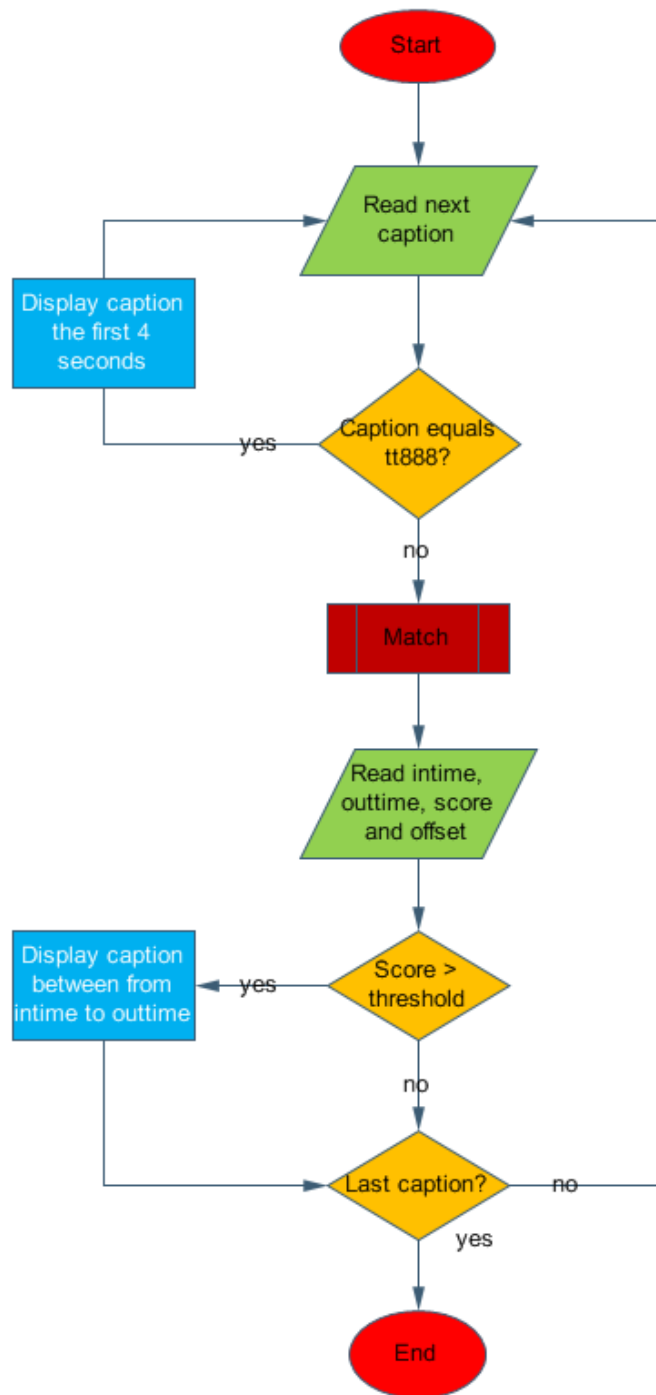


Figure 6.5: *First Pass Alignment*

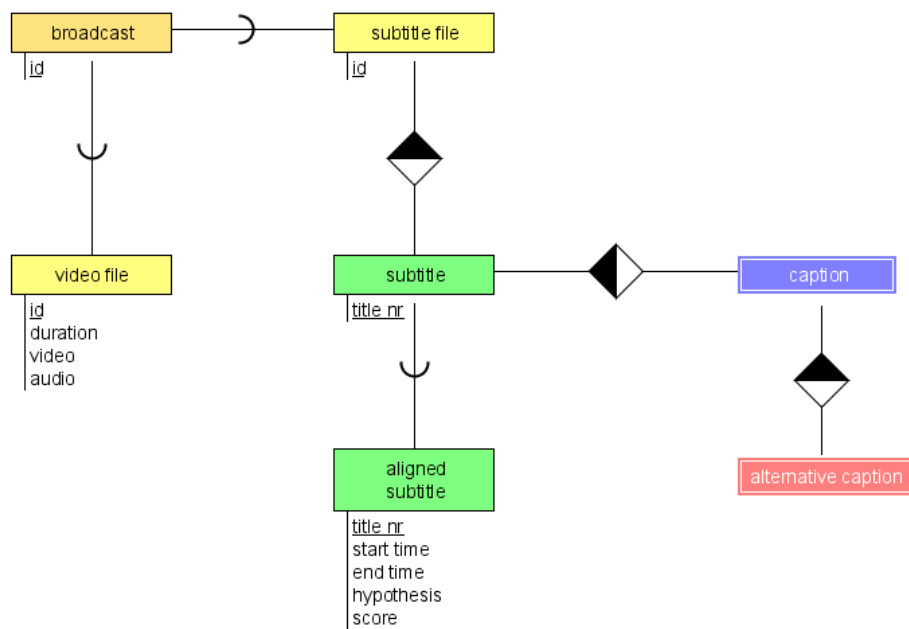


Figure 6.6: ERD extended with entity "alternative caption"

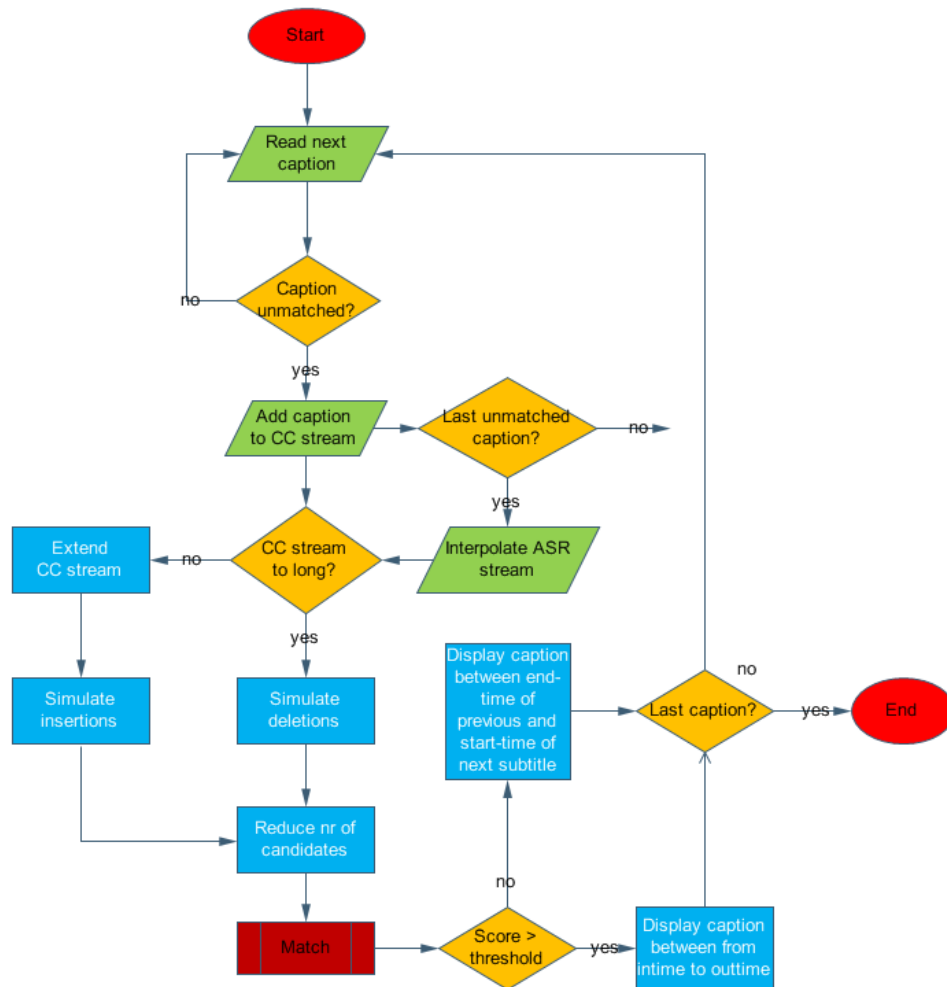


Figure 6.7: *Second Pass Alignment*

Part III

Implementation

In chapter 5 the behavior of SETH was presented i.e. *what* SETH is supposed to do. This chapter discusses *how* SETH is implemented to satisfy the requirements. As the functionality can be broken apart into (sub)processes, the software can be distributed into separate software components each implementing a specific behavior of the system. The decomposition comprises a separate module for each of the states defined in section 6.2.

A global deployment diagram of SETH can be seen in figure 7.1. The blocks represent software components while the arrows show how the information flows through the system and between the components. The components are colored differently to indicate the contributions of the author in the development of a component. Blue indicates that the component is retrieved from other parties but needs to be configured for the system. Green indicates a component that has been developed or configured by the author in collaboration with other employers of Dutcheer. The red components are fully developed by the author and implement the most important part of the research described in this thesis. The shape of a component indicates the nature; round angles indicate the source-files of the subsystem *Npobuilder*, a cubic shape refers to executable files, 3D-bars represent servers. A cube indicates an off-the-shelf component i.e. software supplied by other parties. Usually, such components provide an API and can therefore be integrated in a pipes and filters architecture like the speech model pipeline (section 6.3.1). Round angled blocks represent modules that are directly be invoked by the interpreter. Using off-the-shelf software can save development time but is less flexible since changes usually require recompiling the source code which is not always released.

The interface between SETH and its terminators, as shown in figure 5.1 is implemented by a web application which is described in section 8.7. The user input (flow 1) enters *Npobuilder* at the import module (section 8.2) which extracts and stores the captions and title numbers in a data structure. The data structure (flow I) is passed to the train module (section 8.3). Module train creates speech recognition models (flow II) for the subsequent module named test (section 8.4). The output of the speech recognizer (flow III) is forwarded to the align module (section 8.5). The alignment results are (flow IV) used by the export module (section 8.2) to create a new subtitle file (flow 6) which is returned to the user (flow 2). Finally all multimedia, process, and log files are put together in an archive by the clean module (section 8.6) and available for operators (flow 3) which may decide to upload some improvements (flow 4).

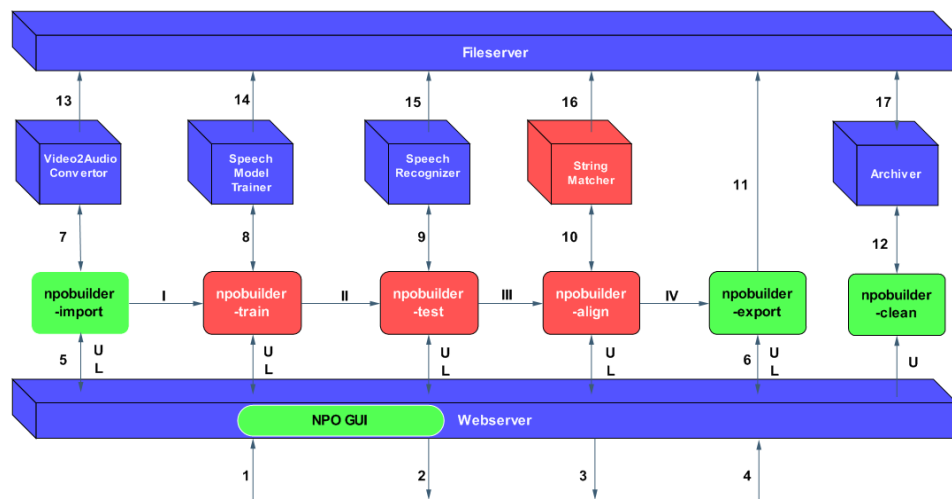


Figure 7.1: *Data flows SETH*

7.1 Video2Audio Converter

Audio file headers contain meta-information such as recording conditions and sampling rate and vary in size. The SONIC speech recognizer assumes that the input audio is header-less (raw), mono, 16-bit and in linear PCM format [Pellom04]. For extracting the audio stream from the video stream, the program FFmpeg [FFmpeg] will be used which is a complete, cross-platform solution to record, convert and stream audio and video. With FFmpeg, an MPEG videofile can be quickly converted to a 16-bit wav-file. From this wav-file, the header is stripped by SoX [SoX]. Both FFmpeg and SoX are freely available off-the-shelf tools. FFmpeg reads the MPG-file (flow 7) as input (parameter -i) and maps the audio content in a 16 kHz (-ar 16000) wav file (-f wav). Next SoX reads the wav file to determine the bitrate and number of channels and converts it into a mono (-c 1) 16-bit (-t .s2) raw audio-file. For analyzing purposes, all media files are stored at the file-server (flow 13). To check the correctness of the video to raw audio conversion, the raw audio file can be imported with an audio editor e.g. Audacity [Audacity] as displayed in figure 7.2.

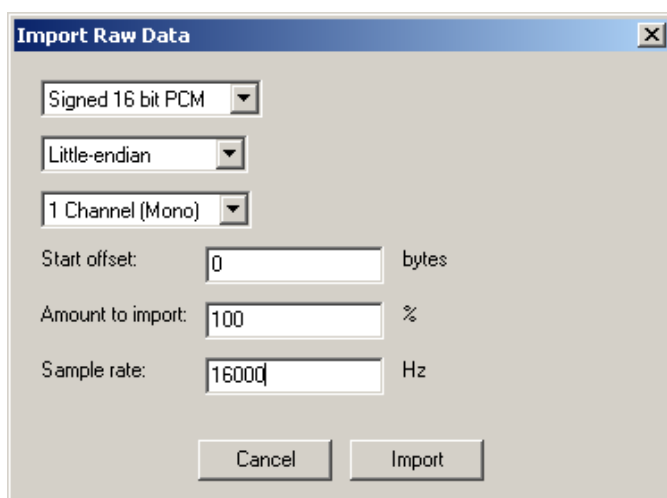


Figure 7.2: *Import raw audio in Audacity*

```
cat $outdir/sentences.txt | text2wfreq | wfreq2vocab >
$vocab cat $outdir/sentences.txt | text2idngram -n 3 -
vocab $vocab -buffer $memory -temp $tmp > $outdir/dig-
its.id3gram
```

Figure 7.3: *Example of activation of CMU-CAM filters*

7.2 Speech Model Trainer

For the construction (training) of language models, the CMU/Cambridge Statistical Language Modeling Toolkit is used, which is a suite of Linux filters that interact with each other and Linux shell scripts by means of pipes as shown in textbox 7.3. The toolkit provides smoothing algorithms including Good-Turing discounting. For the construction of lexicons, a phonetic transcriber called "Autotrans" is used that has been developed in collaboration with TNO. Autotrans is a MS-DOS filter which converts a list of (orthographic) words into phonetic transcriptions. Autotrans is integrated in the Linux environment using DOSEMU [DOSEMU] and tofrodos [Tofrodos].

7.3 Speech Recognizer

The speech recognizer used in SETH is SONIC. SONIC is a HMM based LVCSR platform developed at the University of Colorado. SONIC achieves state-of-the-art performance on a number of commonly evaluated speech databases while also being computationally efficient, real-time, and rapidly portable to new languages. Sonic provides a batch mode in which it processes a predefined list of audio files using a predefined configuration. The batch mode can be invoked via an API.

7.4 String Matcher

The string matcher implements the sliding window algorithm (section 6.4.1). It is written in Haskell and compiled with the GHC compiler [GHC]. Haskell is an advanced purely functional programming language. In particular, it is a purely functional programming language, quite different from most other programming languages [Thompson99]. Functional programs are also relatively easy to maintain, because the code is shorter, clearer, and the rigorous control of side effects eliminates a huge class of unforeseen interactions. With strong support for integration with other languages, built-in concurrency and parallelism, debuggers, profilers, rich


```
makeScore :: [Float] -> Float
makeScore[] = 0
makeScore(x : xs) = x + makeScore xs
```

Figure 7.4: Haskell implementation of equation 6.3

```
7za -t7z foo.7zfoo
svnaddfoo.foo.7z
svncommitfoo.7z -m"archivefoo.7zadded"
rmfoo.7z
```

Figure 7.5: Archiving folder foo

libraries and an active community, Haskell makes it easier to produce flexible, maintainable high-quality software [Haskell]. An example of output produced by string matcher can be found in appendix D. The word-by-word comparison of a candidate ASR string and a caption results in a vector of matches weighted by equation 6.2. The total score is determined by summing the individual scores (equation 6.3). The Haskell code for calculating the total score can be seen in figure 7.4. The input variable [Float] represents a vector of scores.

7.5 Archiver

The archiver component provides the compression, storage and revision control of files. As can be seen in the textbox in figure 7.5, archiver is a pipeline of 7-zip and subversion. The file compression tool 7zip supports several different data compression, encryption and pre-processing filters [7zip]. Subversion is a version control system to manage changes to documents, programs, and other information [Subversion].

The main component of SETH is Npobuilder. Npobuilder communicates both the web server layer and the other components. This includes the conversion between different file formats (flows 5 and 6), the processing of phonetic transcriptions (flow 8) and reading and writing files at the input and output of the speech recognizer respectively (flow 9). Each module can be updated (flow U) via a subversion repository on the web server. The actions of each module (accept clean) are stored in log-files (flow L) on the web server. All modules are written in Perl. The modules are imported into a main Perl program which is invoked with an argument indicating the module name. Npobuilder is executed by a daemon script that checks whether there is data that must be processed (see appendix E. Most of the modules generate Linux shell scripts which are executed by a system call. Since all scripts are archived, the actions can be repeated one after another which facilitates debugging.

8.1 Perl

Perl is a high-level programming language which combines features of the programming language C and the Linux shell commando's *sed*, *awk*, and *sh*. The Perl motto is "There's more than one way to do it" i.e. Perl code can be written in many different ways. Perl supports modular programming i.e. a Perl program can be subdivided into independent reusable pieces of code. A Perl module is a chunk of code that serves as a library to insert into Perl programs to save time and effort [Olson98]. A large collection of Perl libraries and documentation can be found at the Comprehensive Perl Archive Network (CPAN) . The next CPAN Libraries are imported in Npobuilder:

1. Algorithm::Combinatorics
2. POSIX
3. String::Approx
4. Text::Levenshtein

A Regular Expression (RE) is an algebraic notation for characterizing a set of strings for characterizing text sequences. RE's are simple strings much like arithmetic expressions, with a simple and familiar syntax; they are well suited for in describing patterns for string processing RE's can be used for specifying search queries in an Information Retrieval (IR) system (like the WWW) by means of patterns. On a small scale, a RE search will return all sentences that contain the specified pattern. On a wide scale, a RE search will return all documents that contain the pattern.

```

if ($line = /[0-9]+[ ]+[0-9]+[ ]+([a-z]+)/)
{
    print NEW $line;
    push(@asr,$1);
}

```

Figure 8.1: Regular expression to parse time-coded words

The integrated RE text patterns provide powerful pattern matching on strings which makes Perl a flexible text-processing language [Frenz05]. The textbox in figure 8.1 shows a part of the Perl code to write recognized words in a (new) text-file ignoring their corresponding in- and out-frame numbers. The if-statement checks whether or not a line contains a number followed by one or more spaces followed by a number followed by one or more spaces followed by word. If the condition is satisfied, the word is automatically stored in the variable \$1 and pushed in array variable @asr. The RE can be visualized by the FSM in figure 8.2 which has been generated with Graphviz [Graphviz]. The FSM accepts strings like "625120 634880 kader" but rejects e.g. "621120 625120 <SIL>"¹.

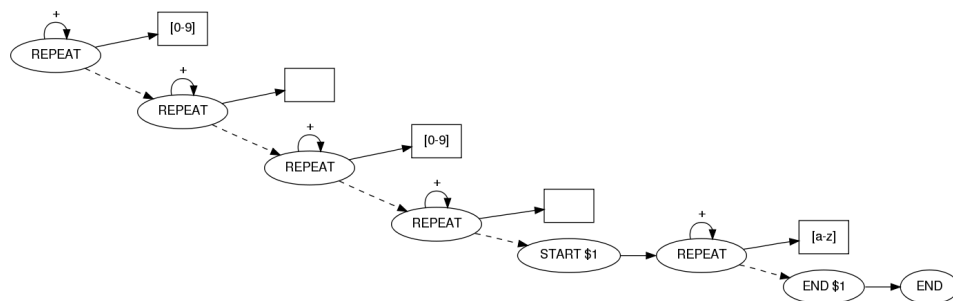


Figure 8.2: FSM to parse time-coded words

¹<SIL> is a filler word representing silence

```
ffmpeg -i foo.mpg -vn -ar 16000 -f wav foo.wav  
sox foo.wav -t .s2 -c 1 -r 16000 -4 foo.raw
```

Figure 8.3: *API calls of FFmpeg and SoX respectively*

8.2 Import and Export Module

The import and export module are responsible for the mapping between the files that are processed by the user and those that are processed by SETH. Files at the input of Npobuilder (flow 5) are expected to satisfy specific predefined file-formats; MPG for video-files and STL for subtitle files. In 5.3 it was stated that only the audio stream of the mpg file is relevant for the speech recognizer. Therefore, the video-file is pushed to the Video2Audio module (flow 7). The textbox in figure 8.3 shows the API calls used to retrieve the required audio format.

Captions and title-numbers are retrieved from the stl-file. Since it has a hexadecimal nature, it can not be opened by a standard editor but requires a special editor like Hexedit [Hexedit] as shown in figure 8.4. Linux has a rich character set and supports all characters being used in a stl-file. So running regular expressions in a Linux terminal should be sufficient to read the textual content of the hexadecimal subtitle. However, we still need to some kind of data structure to store the captions and title numbers such that they can easily be consulted by the system.

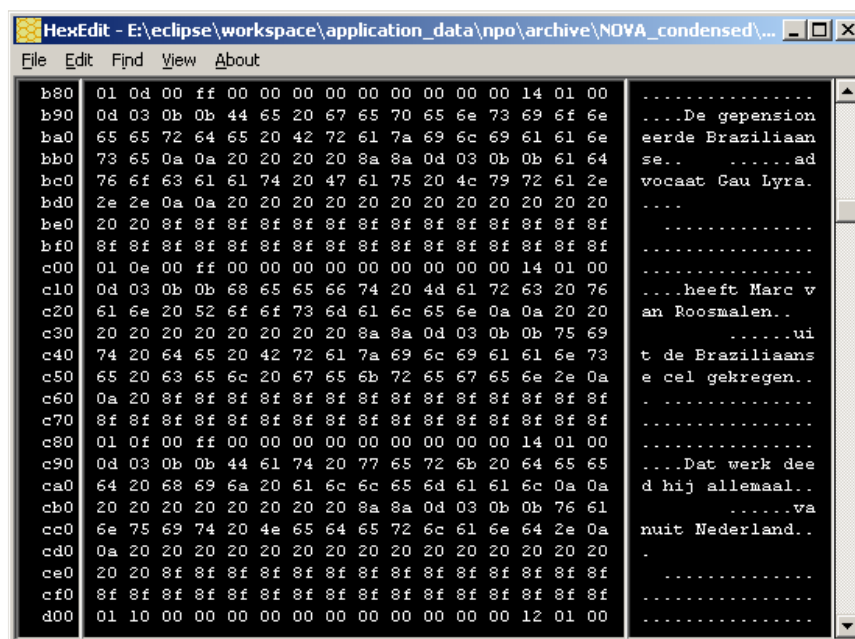


Figure 8.4: *stl* file viewed in Hexedit

8.2.1 XML

XML is an open, portable and well-supported markup language. XML was originally designed to meet the challenges of large-scale electronic publishing but is also an appropriate set of rules to structure application data. It is called an extensible language because users are allowed to define their own mark-up elements [XML]. It is easy to write a parser in a programming language like Perl that can read and write an XML-tree. But XML-files can also be viewed with a browser or text editor. An example of an XML-file containing imported captions can be found in appendix H. The Export Module is responsible for mapping the alignment results from XML to STL and hte publishing both files (flow 6).

```
text2wfreq <foo.norm.txt> foo.wfreq
wfreq2vocab <foo.wfreq> foo.vocab
sort foo.wfreq > temp.wfreq
mv temp.wfreq foo.wfreq
text2idngram -vocab foo.vocab
<foo.norm.txt> foo.idngram
idngram2lm -idngram foo.idngram -vocab foo.vocab
-arpa foo.arpa -context context.ccs
lm_encode foo.arpa foo.lm.bin
```

Figure 8.5: *Shell script for the construction of a language model*

```
cp /foo.dict /home/user/.dosemu/drive_c/lex.dict
echo "Run autotrans"
dosemu
cp /home/user/.dosemu/drive_c/lex.at /foo.at
```

Figure 8.6: *Shell script for the construction of a lexicon*

8.3 Train Module

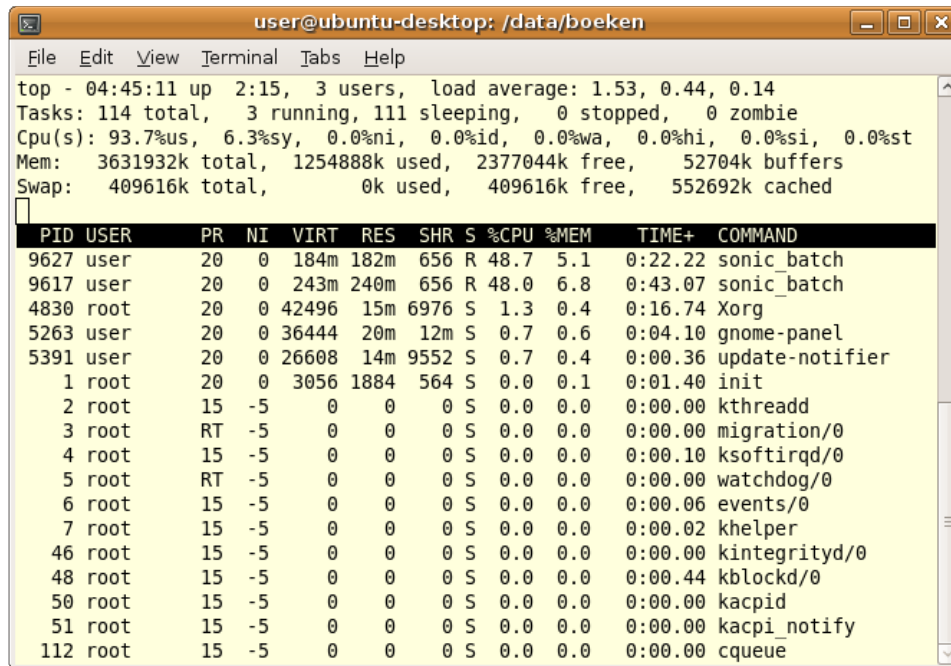
Since the word order in the captions sometimes differ from the word order in the actual speech, Good-Turing smoothing is applied in the construction of each language model. Figures 8.5 and 8.6 display the scripts for the construction of a language model and a lexicon respectively. The captions (flow I) are concatenated to one document which is being cleaned and normalized before being pushed to the speech model trainer (flow 8). An example of a normalized trainfile can be found in appendix F.

```
sonic_batch - cf001.cfg &  
sonic_batch - cf000.cfg
```

Figure 8.7: SONIC API calls

8.4 Test module

The test module is concerned with the actual speech recognition process. The language and pronunciation model (flow II) together with a constant acoustic model are pulled to start the recognition of the speech contained in the audio stream. The acoustic model has been developed in collaboration with TNO. It was trained on the "Corpus Gesproken Nederlands" (CGN). The CGN corpus contains a huge amount of Dutch speech spoken by many different speakers from all over the Netherlands. The name and location of the speech models, log-file, output-file, list of audio files, phonetic alphabet file, speech filler file, etc. are specified in a configuration-file. The configuration includes recognition parameters like e.g. sample-rate, filler-penalty and feature-type. The parameter "align word hypothesis" ensures that for each recognized word the in- and end-frame will be printed. In appendix G an example of a SONIC configuration-file can be found. The SONIC speech recognizer is activated with a simple API call. Figure 8.7 shows calls to run two simultaneous SONIC instances as indicated by the output of the Linux program *top* (figure 8.8).



```

top - 04:45:11 up 2:15, 3 users, load average: 1.53, 0.44, 0.14
Tasks: 114 total, 3 running, 111 sleeping, 0 stopped, 0 zombie
Cpu(s): 93.7%us, 6.3%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3631932k total, 1254888k used, 2377044k free, 52704k buffers
Swap: 409616k total, 0k used, 409616k free, 552692k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
9627	user	20	0	184m	182m	656	R	48.7	5.1	0:22.22	sonic_batch
9617	user	20	0	243m	240m	656	R	48.0	6.8	0:43.07	sonic_batch
4830	root	20	0	42496	15m	6976	S	1.3	0.4	0:16.74	Xorg
5263	user	20	0	36444	20m	12m	S	0.7	0.6	0:04.10	gnome-panel
5391	user	20	0	26608	14m	9552	S	0.7	0.4	0:00.36	update-notifier
1	root	20	0	3056	1884	564	S	0.0	0.1	0:01.40	init
2	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	15	-5	0	0	0	S	0.0	0.0	0:00.10	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	15	-5	0	0	0	S	0.0	0.0	0:00.06	events/0
7	root	15	-5	0	0	0	S	0.0	0.0	0:00.02	khelper
46	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kintegrityd/0
48	root	15	-5	0	0	0	S	0.0	0.0	0:00.44	kblockd/0
50	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
51	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpi_notify
112	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue

Figure 8.8: Two simultaneous SONIC instances

8.5 Align Module

The align module implements the FPA and SPA discussed in section 6.4. It reads the recognition results (flow III), invokes the string matcher (flow 16) and selects the best matching ASR output string (hypothesis). The time-codes (tci and tco), hypothesis, score and any alternative caption are being stored in a XML structure (see appendix H). . Once the XML-tree is complete i.e. all captions are processed, it is pushed to the export module (flow IV).

8.6 Clean Module

After returning the result, SETH is waiting for new input. To save local disk space and keep the workspace tiny, all current input-files, models, log-files, output-files, etc. (flow 13-16) are moved from the workspace to an archive (flow 12) on the file-server (flow 17).

8.7 Web Server

As shown in the context diagram in 5.1, both the user and operator interact with the system. It is decided to host the application at Dutcheer office so the customer must have remote access in order to use the application. This is the reason that SETH is being implemented as a web application. At the client-side SETH is accessible via a web-browser by typing an URL .

Figure 8.9 displays the GUI for submitting a stl-file and mpg-file. The user needs to specify the broadcast name and adjust the reading rate with the sliding bar. When the user submits the form (by pressing button "Uitvoeren", the file-extensions are checked. If the input is not accepted, a warning will be displayed, otherwise the data is being sent to SETH. The progress-bar gives an indication of the remaining upload-time. After a successful transfer, the browser shows the GUI in figure 8.10 which is very useful for monitoring Npobuilder. The polling deamon will activate Npobuilder directly after receiving the input data. The GUI shows the current state of SETH. A process is checked off when one or more files that are produced by the process exist and the preceding process is checked off (see figure 8.11. When all states are checked off, the browser renders a download-page at which the alignment results are available (see figure 8.12).

At the server side, Npobuilder and its peripheral software run on a machine at Dutcheer. Operators do not need a GUI but can directly access the web server with FTP tools like SSH and SCP to maintain and update the system (flow 4). The Web server is implemented by XAMPP [XAMPP] which is an easy to install Apache distribution containing MySQL, PHP and Perl . The web application is generated by PHP . PHP is a widely-used general-purpose scripting language that is especially suited for web applications. JavaScript is used to validate the input before it is being sent to the system by checking the extensions. An extension check is done to check whether the files have the proper format. Video files must be stored in the MPEG format while subtitle files are stored in stl-files . An stl-file is an EBU-standard open and teletext file in which captions, title numbers, time codes, etc. are stored. Ubuntu Desktop 8.10 is chosen as operating system because it is robust, free available and offers a lot of free programming tools. To enhance portability, easy backup and low consuming of physical resources the operating system runs as a Virtual Machine (VM) on a server [VMware] with a link to the file-server.

[Download](#) [Log uit](#)

SPEECH ESTIMATING TITLE HEURISTICS

Voer onderstaande gegevens in

Titel van de uitzending

File naam STL

File naam video (MPEG)

Leessnelheid (aantal woorden per minuut)

152

145 200

Alignment Tool voor ondertiteling - Dutchear / NPO 2009©

Figure 8.9: Form for submitting a stl-file and mpg-file

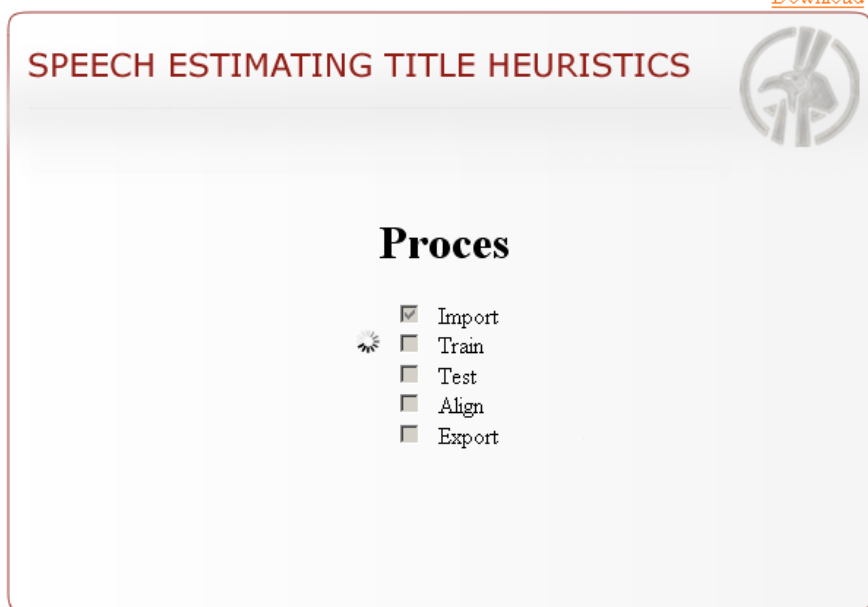


Figure 8.10: *GUI for processes monitoring*

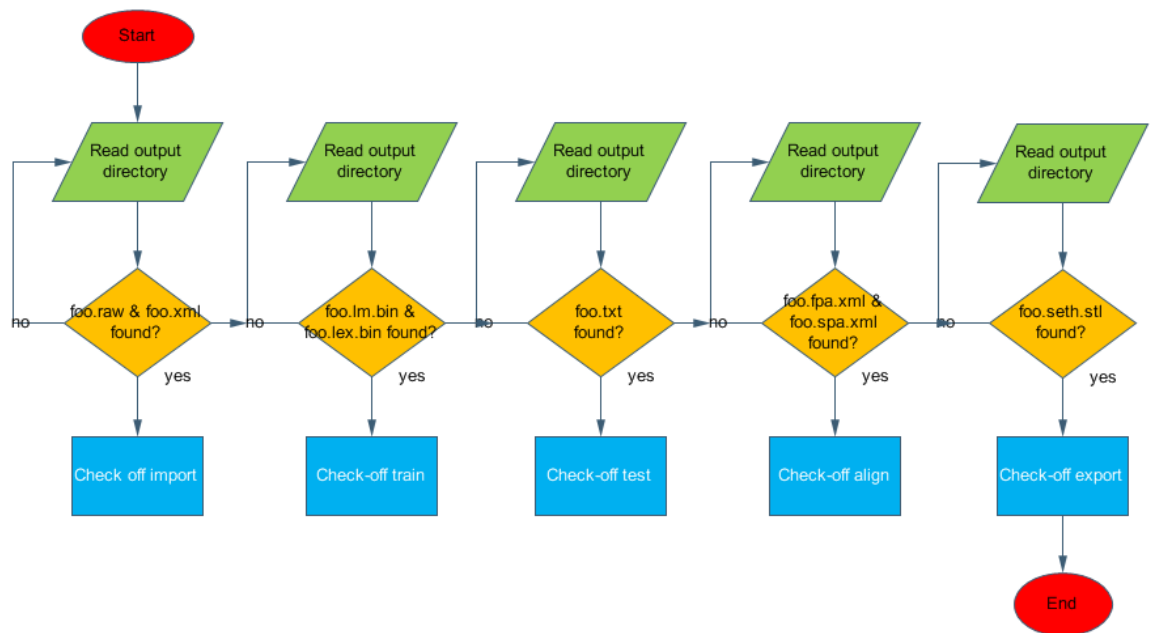


Figure 8.11: *Flowchart for determining the progress*

[Download](#) [Log uit](#)

SPEECH ESTIMATING TITLE HEURISTICS



Download

[nieuws.fpa.xml](#)
[nieuws.seth.stl](#)
[nl_helpt_16dec.fpa.xml](#)
[nl_helpt_16dec.seth.stl](#)
[keuring_16dec.fpa.xml](#)
[keuring_16dec.seth.stl](#)
[rail_away_run_4.fpa.xml](#)
[rail_away_run_4.seth.stl](#)

[Uploaden](#)

Figure 8.12: Download-page for the new subtitle-file

Part IV

**Results, Conclusions and
Recommendations**

The performance of SETH is being tested on the data described in chapter 4. The accuracy is measured by counting the number of ASR-CC matches with a score > 8.1 bit. Table 9.1 presents the percentage of successfully aligned captions. All broadcasts were aligned with the default value of the reading rate (145 words per minute). Table 9.2 shows the differences in both the alignment and execution performance using an universal LM and lexicon.

Table 9.1: *Alignment results*

broadcast	first pass alignment	second pass alignment
Boeken	23%	24%
Iedereen kan aquarelschilderen	80%	80%
De vloer op	25%	25%
NOVA condensed	70%	71%
NOVA verbatim	70%	71%
t Vrije schaep	15%	15%

Table 9.2: *Performance difference using universal speech models*

broadcast	first pass alignment	second pass alignment	max ASR real time factor
Boeken	-2 %	-2%	-1.30
Iedereen kan aquarelschilderen	-34%	-29%	-1.35
De vloer op	-6%	-5%	-1.65
NOVA condensed	-25%	-26%	-1.60
NOVA verbatim	-22%	-22%	-1.68
't Vrije schaep	-7% %	-7%	-2.13

Only "Iedereen kan aquarelschilderen" and "NOVA" have a performance above 50%. The speech in both broadcasts is rather formal. However, "Boeken", which has a similar kind of speaking style, scores below 50%. Apparently, the English words and crosstalk have a large impact on the performance. The less accurate is the alignment of "De vloer op" and "'t Vrije schaep". The speech speaking style in both broadcasts is spontaneous and very informal and both recordings contain background noise. The performance difference between the NOVA condensed and NOVA verbatim is small probably because both word identity weighting and text reduction concentrate on *important* words which are harder to recognize with a universal language model and lexicon (see table 9.2). There is no big difference between FPA and SPA which suggests that the key to a better performance is not a further optimization of the alignment algorithm but a better speech recognition performance. The effort to generate new speech models for every new broadcast is justified by the decrease in both accuracy and processing rate (higher real-time factor) when using universal models. Again the influence of speech recognition on the overall performance is significant.

10.1 Evaluation of Activities

There was little direct literature on this topic. The data examination was very useful; the small dataset has already shown a large number of difficulties. The meetings with the customer were also very fruitful. The main part of the development time is spent at providing the fully automatic on-line service of SETH at the expense of experiments and tuning. The architectural design could also be used for an audio retrieval application (see appendix A).

10.2 Discussion

The design allows the reuse of filters from off-the-shelf toolkits and Linux shell commands. It is possible to run two simultaneous instances of the speech recognizer. The average percentage of successful aligned titles amounts 47%. The overall execution time approximates 1.2x real-time i.e. it takes less than 5 quarters of an hour to process a 1 hour during broadcast. The experiments show that the proposed software design is very suitable to build an automatic closed caption alignment system. The possibility to run individual modules in isolation provides a friendly means for tuning. If the text used for training the speech recognizer reflects the vocabulary used in the specific broadcast, much better speech recognition results are achieved than using general models. The best results are achieved when the speech is rather formal and non-spontaneous, pure Dutch speech pronounced by a native speaker and does not contain crosstalk nor background noise. Dissimilarities between the speech and captions are not a major problem as long as the captions include the most important words. The alignment algorithm is also robust to most of the insertions caused by music. Deviant language use, background noise, spontaneous speech, strong regional accents are still a difficult job for the speech recognizer and hence a major problem in automatic closed caption alignment. The key to a better performance is to find a means to increase the speech recognition performance. Since there will always be broadcasts with poor speech quality, manual verification or adaptation of the subtitles remain necessary.

101

11 RECOMMENDATIONS FOR FUTURE WORK

It is recommended to focus on speech recognition quality. Perhaps the speech recognition accuracy can be increased by using grammars instead of SLM's. It also pays to find a means to suppress background noise and music. Both the alignment and speech recognition might be more accurate when local rather than global word counts are being used. Local word counts can be computed by a rough partitioning of the titles over time intervals of the audio track. The evaluation method can be advanced by constructing a histogram of the in- and out-time differences between automatically and manually aligned captions. Regarding the implementation, a better integration might be possible in a Java environment using Tomcat [Tomcat] and Xuggler [Xuggler]. 103

Appendices

A RESEARCH PAPER

This appendix comprises the research paper that was submitted for the master course "System Specification Models (IN4091)" which is consistent with the research described in this thesis and my daily R&D work at Dutcheer.

Architectural Design for Speech Applications

System Specification Models

Jeroen Boogaard (1139711)
Jeroen.boogaard@gmail.com

Delft University of Technology

Abstract

The development of speech applications is usually very complex as it involves a lot of data processing in dynamic environments and speech recognizers from different manufactures require a different integration. A good software architecture design reduces both the technological and organizational complexity. This paper provides an analysis of the processes involved in a speech application. The processes are assigned to components composed by lower level modules. The problem to be solved is how to design the architecture of the speech application such that it permits both the reuse and replacement of (off-the-shelf) tools and allows understanding and communication at a high level? The solution is sought in a pipes and filters based architectural design. An Automatic Speech Recognition (ASR) pipeline is presented that specifies the function and allowed data types of filters at different levels. The design is evaluated by testing two speech applications in which the ASR pipeline is implemented.

1 Introduction

The huge amount of speech containing multi media like movies, filmed meetings, recorded telephone calls, etc. demand increased availability like the ability to search for specific speech fragments [1] or subtitle attachment [2]. Speech applications provide this kind of functionality by performing ASR to determine what is being said and at what time by mapping recorded speech to time coded text. The development of speech applications is usually very complex as it involves a lot of data processing in dynamic environments. Speech recognizers from different manufacturers differ in the layout of recognition output, supported audio formats, phonetic alphabet being used, etc.

and therefore require a different integration. To keep both the costs and complexity of the speech application development process within bounds, software architectural design choices are very important. The better the design, the more understandable and maintainable the resulting software will be.

1.1 Architectural Design

Architectural design is concerned with decomposition *into*, and specification of the interaction *between* individual system components. A good software architecture provides an abstract system view that allows designers and developers to reason about system integrity constraints and overall system behavior [3]. Monroe et al. [4] enumerates the following benefits:

- * reuse of both design and code is promoted
- * increased understandability e.g. by hiding details
- * style specific analysis by constraining the design space
- * style specific graphical depictions of design elements

An example of an architectural style is the pipes and filters architecture. The pipes and filters architecture defines a system as a collection of computational filters. The filters are connected by pipes that provide the data transmission between the output of a filter and the input of the next filter. Each filter provides a complete transformation from input to output. The pipes and filters together form a pipeline. An example is the rendering pipeline in computer graphics [5]. More about the pipes and filters architecture can be found in section 2.

As a software engineer of speech applications, one is often confronted with a number of complexity issues:

1. complexity inherent to speech recognition (different speaking styles, data sparseness, etc)
2. complexity inherent to natural language processing (ambiguity, homonyms, etc)
3. integration of off-the-shelf software
4. the need for regular model updating
5. future plans for extending or enhancing the software
6. concurrency and synchronization
7. source code in different programming languages
8. time scheduling and resource management
9. communication mechanisms

The first two complexity issues are related to research activities like literature study and experiments. The other issues are rather architectural design concerns [6]. Malan et al. [7] distinguishes two intractability categories:

- * Intellectual intractability:
the technological complexity that may arise from novelty, dependencies technologies employed, etc.
- * Management intractability:
the organizational complexity that lies in building the system by a project team

Speech application issues that fall into the first category are 3-6 while 7-9 are typical management intractability issues.

1.2 Structure of the paper

This paper provides a proof of concept of an architectural design for the integration of speech technology in software applications. The amount of data processing involved in speech applications motivates a study of the pipes and filters architectural design in section 2. In section 3 requirements needed for a flexible integration of ASR in a speech application can be found. Section 4 presents a global design of speech applications. In section 5, a top-down design of the ASR pipeline can be found. The flexibility of the design is discussed in design in section 6. Experiments are described in section 7 and the conclusions can be found in 8.

2 Pipes and Filters Architecture

The pipes and filters architectural design can be described by the analogy of a public water system. Water flows through a water pipe to a filter that adds something to or take something away from the water. Filters can be easily changed without upsetting the overall system as long as the input and output can still be streamed through the pipes. The water pipes correspond to data pipes and filters correspond to data transformations [8].

In the pipes and filters architecture, the system task is divided into sequential processing steps each separately implemented as a filter. Filters are connected by data pipes to form a processing pipeline. Each filter consumes data at its input and provides the transformed data at its output. *Active* filters actively push or pull data through a pipeline. Active filters can run in parallel since they are able to start to work with the partial results of its predecessors instead of having to wait for their completion [9]. Passive filters are activated by receiving pushed data, or by receiving a pull request. Pipes connect a filter (upstream process) output to the next filter (downstream process) input. Pipes do not transform data but function as uniform interconnection buffers. Special kinds of pipes are typed pipes, which require that the data passed between two filters have a well-defined type. The input to the processing pipeline is provided by a data source such as a text file while the output flows into a data sink such as a file or terminal. A pipeline can be subdivided into filter chains, each responsible for a specific task.

The pipes and filters architectural style allows designers to understand the overall input/output behavior of a system as a composition of the individual filters. Filters can easily be added or updated and reuse is encouraged. Concurrent execution is possible by running filters on a multiprocessor system. Linux operating systems provide a uniform pipe mechanism and a lot of tools can be combined in any way with a simple shell command. The pipes and filters architectural style thus supports understandability, maintainability, concurrency and re-usability. A disadvantage is

that this architectural style is not really suitable for interactive systems.

3 Requirements

In speech applications, a speech recognizer is provided a stored or streamed audio signal and returns transcribed utterances and their corresponding time slots. As can be seen in figure 1, a speech recognizer performs a feature extraction to the speech signal to produce an observation sequence for which the most likely sentence is reproduced by an optimization/search process [10, 11]. This process requires three, language dependent, (statistical) speech models:

- * a pronunciation model or lexicon representing words by phonetic pronunciations
- * one or more language models to estimate the most likely word sequence
- * an acoustic model to estimate the most likely observation sequence given an utterance

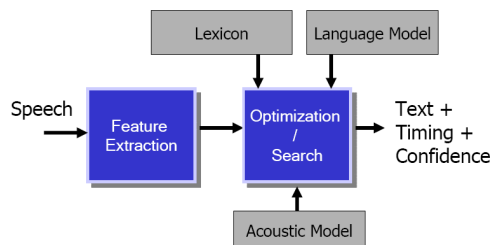


Fig. 1. Overview of automatic speech recognition (<http://www.cis.hut.fi/Opinnot/T-61.184/>)

A sufficient ASR quality requires that the models that are up to date and relevant to the application domain [12]. Consequently, the speech application architecture must provide a means, at least manual, for regular model updating i.e. training on newly available training data. Another architectural requirement is that for testing purposes, it must be possible to isolate the speech recognizer from the other components.

There are both research and commercial motivations to even allow a complete replacement of speech recognizer in a speech application. Since the ASR process heavily claims the available hardware resources, it could be necessary to run the speech recognizer on a separate processor. Therefore, the speech recognizer component must always remain a sufficient level of independence when inter-weaved into the application.

4 System Overview

In the introduction, two examples of speech applications were mentioned; audio retrieval and automatic subtitle alignment. Both applications use the time coded text to get grip on the speech content in a multi media stream. However the diagram displayed in figure 2 applies to any system that performs artificial intelligent techniques on spoken content.

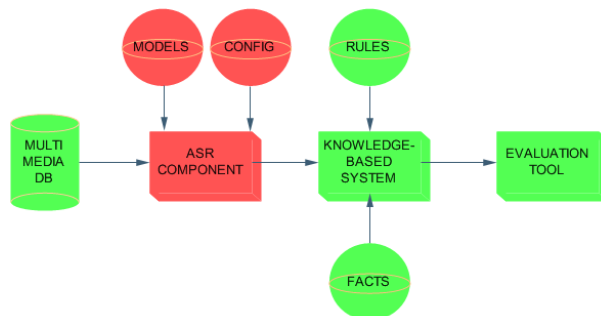


Fig. 2. Global design of a speech application

The diagram shows that an ASR component pulls a multi media stream from a multi media database. The ASR component represents a non-monolithic subsystem containing the speech recognizer as delivered by a manufacturer as well as all functionality that allow the integration into a speech application. The multi media database represents any data source that can provide an audio/video flow e.g. radio journal, YouTube movie, etc.

Since there is no control process between the multi media database and the ASR component, the latter does not have any *a priori* knowledge about the nature of the multi media stream that it must process. The ASR component also pulls the three speech models described in section 3. Finally, a configuration file (config) is being pulled that prescribes speech engine parameters like feature extraction and search space options.

After a (successful) speech recognition, the results are pushed to a knowledge-based system which is responsible for the main service of the speech application. In knowledge-based system, the time coded text is combined with other facts which results in a specific action driven by production rules [13]. The evaluation tool presents any component that reports about the quality of the system by reading the output of the knowledge base.

Note that the ASR component does not receive any feedback from the knowledge-based system. The underlying idea is that it should not matter how the recognition results are further processed or what the nature of the knowledge-based system is. Consequently, any component succeeding the ASR component is independent of the speech recognizer being used which implies that the time coded text must always be represented with a predefined layout.

5 ASR Pipeline

In the global design of the previous section, the speech recognizer was presented as some kind of black box (ASR component) that can handle any multi media file and always return its results in the same layout. The design satisfies the independent criterion discussed in section 3. This section zooms in on the ASR component and describes its functionality in a top-down pipes and filters architectural design. In the diagrams, the names of the off-the-shelf filters are written in *italics*.

Speech recognition can be regarded as a sequence of conversions from multi media file to time coded



Fig. 3. ASR component represented by a pipeline

text using statistical speech models that are generated by training data. Pellom [14] shows that the generation of a lexicon and language model for the SONIC speech recognizer is also a sequence of transformations. Therefore, the design in figure 3 shows the ASR component as a pipeline. The ASR pipeline comprises the connection between the speech model pipeline and the signal processing filter chain.

component	function	input type	output type
speech model pipeline	make speech models	document	binary
signal processing filter chain	convert speech to text	multi media	xml

Table 1. Specification ASR Pipeline

In table 1, the function and allowed data types are specified. Input type 'document' indicates that any text layout is allowed for the creation of the lexicon and language model. The type 'binary' indicates a speech recognizer specific encoding. XML is chosen as the universal layout at the output of the ASR pipeline.

5.1 Signal Processing Filter Chain

The signal processing filter chain represents all transformations from a multi media file to an XML file. As can be seen in figure 4, the signal is piped through filters *FFmpeg* [15] and *SOX* [16] respectively, before it reaches the SONIC speech recognizer. This is because SONIC assumes that the audio at the input is "header-less", "mono", 16 KHz, 16-bit and in linear PCM format [18]. Similar constraints hold for speech

recognizers other than SONIC. The diagram shows how the filter chain is composed by off-the-shelf tools (FFmpeg, SoX and SONIC). A developer has only to develop text2xml.

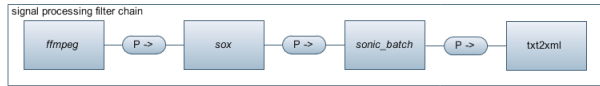


Fig. 4. Signal processing filter chain

As can be seen in the specifications in table 2, the FFmpeg filter permits the multi media at the input of the filter chains to be of any type while the text2xml filter restricts the output to XML.

filter	function	input type	output type
ffmpeg	extract audio from multi media	multi media	wav
sox	strip header information	wav	raw
sonic_batch	convert speech to text	raw	ascii
txt2xml	extract words and time slots	ascii	xml

Table 2. Specification signal processing filter chain

As described in section 3, a speech recognizer needs an acoustic, language and pronunciation model. To keep the language and pronunciation model up to date with respect to the application domain, it is necessary to regularly retrain them on newly available text documents. Table 1 specifies that are no restrictions on the type of document being used for training. The impact of this higher level design decision is the need for a train-file filter chain at a lower level. The train-file filter chain should always produce a valid train-file regardless of the type of documents received.

5.2 Speech Model Pipeline

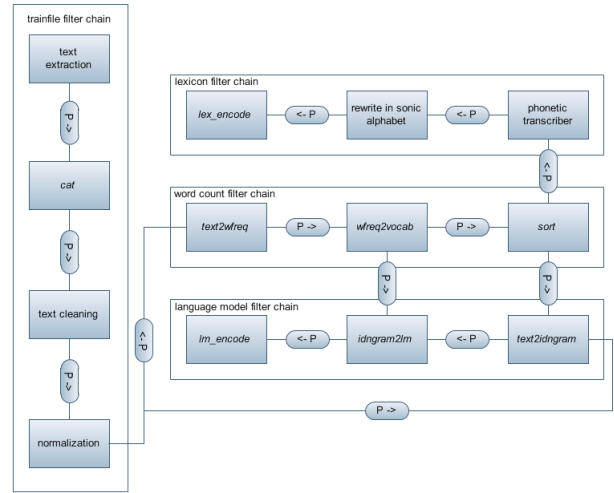


Fig. 5. Speech model pipeline

According to the ASR pipeline specification, the output of the speech model pipeline is binary which means that the speech models must be encoded into binary files that are compatible with the speech recognizer software. As can be seen in figure 5, these two binary speech models are produced by two separated filter chains at the end of the pipeline. Since they do not depend on each other, the lexicon and language model can be created simultaneously. The pronunciation and language model filter chain share a common word count filter chain. The specification of the filter chains as components of the speech model pipeline can be found in table 3.

The train-file filter chain comprises a number of sequential text transformations to represent the lingual context of the application system in a train-file. The support of different document types implies text layout conversions including Optical Character Recognition (OCR). All text resources are collected by the "cat" filter, a readily available program in Unix systems. Text cleaning involves the conversion

filter chain	function	input type	output type
train-file filter chain	create train-file	document	ascii
word count filter chain	make statistics	ascii	ascii
lexicon filter chain	make lexicon	ascii	bin
language model filter chain	make language model	ascii	bin

Table 3. Specification speech model pipeline

of upper-case to lower-case characters and the removal of punctuation marks and unwanted white-spaces. Text cleaning is necessary to avoid that different occurrences of the same word are regarded as different words [19]. The text cleaning filter also replaces digits by words. The normalization filter adds structure to the text like sentence and paragraph tags. The word count filter chain implements filters from

filter	function	input type	output type
text extraction	extract text from documents	document	ascii
cat	compose text stream	ascii	ascii
text cleaning	make proper layout	ascii	ascii
normalization	add structure	ascii	ascii

Table 4. Specification train-file filter chain

the CMU/Cambridge Statistical Language Modeling Toolkit [20], a toolkit to create language models. The specification in table 5 shows that filter wfreq2vocab transforms word frequencies to a vocabulary. As can be seen in the diagram in figure 5, the vocabulary can be reused in the lexicon filter chain.

filter	function	input type	output type
text2wfreq	determine word frequencies	ascii	wfreq
wfreq2vocab	make vocabulary	wfreq	vocab
sort	sort vocabulary	vocab	vocab

Table 5. Specification word count filter chain

The idea behind the lexicon filter chain, specified in table 6 is to provide a means for automatic lexicon creation in different phonetic notations. The design allows changes in the phonetic representation to be performed in a conversion filter (rewrite in sonic alphabet) leaving the phonetic transcriber unchanged. The conversion filter is also responsible that the lexicon satisfies the layout as being specified by the speech recognizer tool that integrates the lexicon (lex.encode).

filter	function	input type	output type
phonetic transcriber	make phonetic transcriptions	vocab	lex
rewrite in sonic alphabet	convert phonetic transcriptions	lex	lex
lex.encode	make binary lexicon	lex	bin

Table 6. Specification lexicon filter chain

Like the signal processing filter chain, the language model filter chain only contains off-the-shelf filters which means that only the pipes have to be developed. Table 8 shows the off-the-shelf filters being used in this design and their origin. The key concept is that the design indicates which building blocks need to be developed and how they can be combined with existing tools.

filter	function	input type	output type
text2idngram	make n-grams	ascii & vocab	idngram
idngram2lm	make language model	vocab & idngram	arpa
lm.encode	make binary language model	arpa	bin

Table 7. Specification language model filter chain

filter	origin
ffmpeg	web
sox	web
sonic_batch	sonic
cat	linux command
text2wfreq	cmu-cam toolkit
wfreq2vocab	cmu-cam toolkit
sort	linux command
lex.encode	sonic
text2idngram	cmu-cam toolkit
idngram2lm	cmu-cam toolkit
lm.encode	sonic

Table 8. Origin off-the-shelf filters

6 Flexibility

The flexibility of the ASR pipeline can be understood by considering a number of realistic requirement changes. Assume that instead of SONIC, SPRAAK [17] is used for speech recognition. How well does the proposed design allow the replacement of the speech recognizer? Different speech recognizers often use a different phonetic alphabet which implies also a different acoustic model. In the signal processing filter chain, only `sonic_batch` is replaced by another filter.

Since the duration of the speech recognition process is near real time, the ASR process is usually the most time consuming process. How can parallel processing be applied to decrease the processing time? The filters in the signal processing filter are sequentially chained which means that they have to wait for each other. Although a streaming process will permit some kind of parallel processing, the conversion from speech to

text will dramatically slow down the entire process. The most effective, but hardware intensive, solution is to split the media file into audio fragments and process two or more of these fragments simultaneously. For this, the signal processing filter chain must be modified such that it contains a number of redundant `sonic_batch` filters. The `text2xml` filter must pull multiple ASCII streams and present the recognition results chronologically in the XML stream.

7 Experiments and Results

Two ASR pipeline has been implemented in two different speech applications; an audio retrieval system and a system for the automatic alignment of subtitle captions. These speech applications use the time coded text to get grip on the speech content in a multi media stream. Both applications were deployed according to the diagram in figure 4.

The goal of audio retrieval is to find relevant speech fragments given a typed query. Therefore, the audio retrieval system was evaluated by the amount of correctly returned fragments i.e. media time stamps determined by the speech recognizer, given a query. The multimedia database contained two different recordings; a council and a lawsuit. For both recordings relevant text documents have been collected as input to the train-file filter chain to generate a language model and lexicon. The council recording is a video file while the lawsuit recording contains only audio. Both media files were successfully converted to a suitable audio stream for the speech recognizer and next to a universal XML document by the signal processing chain. Two different speech recognizers were implemented SONIC and Autonomy Softsound. For SONIC the presented design could be used exactly while Softsound required a different instance in the train-file filter chain and signal processing filter chain i.e. replacement of components that have a comparable task. A list of search queries with the corresponding times stamps was manually created. The queries were fired to the knowledge base and the results were compared with the manual assigned time stamps.

	sonic	softsound
council	37%	61%
lawsuit	59%	79%

Table 9. Results of audio retrieval

Table 9 shows the retrieval for the different recordings and speech recognizers. The table shows that most of the retrieval rates are above 50%. The design is thus suitable to be used for the development of this kind of speech applications. The low score of 37% is due to the configuration of the speech recognizer. To improve the scores, the ASR pipeline provides a clear component structure to change the overall effect of small changes in the configuration. For example, one could measure the effect of adding more training texts. Having a suitable architecture, most of the future effort will be focused on research on speech recognition rather than on software development issues.

The problem of automatic subtitle alignment is to assign a predefined set of captions a start time corresponding to the lip movements. The time codes are to be found by aligning each caption with a substring of the speech recognizer output. The evaluation comprises number of captions that could be matched with fragments of the speech recognizer output. In the experiment an educational television program about how to make aquarel paintings was used. The subtitle captions were used by the train-file filter chain to generate a relevant language model and lexicon. The video stream was converted to a suitable audio stream for the SONIC speech recognizer. The knowledge base contains an experimental alignment algorithm which could match 69% of the captions. The experiment shows that the ASR pipeline can also be used for integrating speech recognition in an automatic caption alignment system. The performance can be increased by improved alignment techniques. The architecture allows an isolated treatment of the knowledge base without any changes to the ASR pipeline.

8 Conclusions

The pipes and filters architecture fits the dynamic and data intensive speech application domain by defining all data flows and regarding component at each abstraction level as simple transformations. The design allows the reuse of filters from off-the-shelf toolkits and Linux shell commands. The experiments show that the ASR pipeline can be successfully used to build speech applications. In the future, experiments with the parallel execution of redundant filter chains will be done.

References

1. J. Foote
An Overview of Audio Information Retrieval
Institute of Systems Science, 1997
2. C. Huang
Automatic Closed Caption Alignment
Colombia, 2003
3. S. A. White
Software Architecture Design Domain
University of Houston, 1996
4. Monroe, Kompanek, Melton, Garlan
Stylized Architecture, Design Patterns, and Objects
1996
5. T. Funkhouser
3D Polygon Rendering Pipeline
Princeton University, 2000
6. B.D. Appleton
Software Design Specification
Construx Software Builders, 1997
7. R. Malan and D. Bredemeyer
Software Architecture Action Guide
Bredemeyer Consulting, 2009
8. E. Gamma, R. Helm, R. Johnson and J. Vlissides
Design Patterns - Elements of Reusable Object-Oriented Software
Addison-Wesley, 1995
9. A. Langhorst and M. Steinle
Pipes and Filters Architectural Pattern
Hasso-Plattner-Institute for Software Systems Engineering, 2003
10. P. van Alphen
HMM-based continuous-speech recognition
Universiteit van Amsterdam, 1992
11. D. Jurafsky and J. H. Martin
Speech and Language Processing, International Edition
Prentice-Hall, 2000
12. P. Wiggers
Modelling Context in Automatic Speech Recognition
Technische Universiteit Delft, 2008
13. M. Negnevitsky
Artificial Intelligence, A Guide to Intelligent Systems
Addison-Wesley, 2002
14. B. Pellom
Sonic: Digit Recognition Tutorial University of Colorado, 2002
15. <http://www.ffmpeg.org/>
FFmpeg
16. <http://sox.sourceforge.net/>
SoX - Sound eXchange
17. <http://www.spraak.org/>
The SPRAAK Speech Recognizer
18. B. Pellom and K. Hacıoglu
SONIC: The University of Colorado Continuous Speech Recognizer
University of Colorado, 2004
19. A. de Klerk
Keyword Identification for Service-Desk Call Classification
2006
20. http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html
The CMU-Cambridge Statistical Language Modeling Toolkit v2

This appendix shows the phonetic transcriptions used for the recognition of the speech contained in the television program "Iedereen kan aquarelschilderen".

orthographic	phonetic
[adem]	adem
[eh]	eh
[ehm]	ehm
< s >	SIL
< /s >	SIL
< UNK >	
SIL	SIL
aan	a : n
absorbeert	ApsOrbe : rt
actuele	Aktyl@
advies	Atfis
adviseren	Atfize : r@n
afstand	AfstAnt
afvegen	Afe : g@n
al	Al
allebei	Al@bEI
allereerst	Al@re : rst
allerlei	Al@rEI
als	Als
altijd	AltEI
ander	And@r
anders	And@rs
ansichtkaartje	AnsIxtka : rtj@
aquarel	a : ka : r@l
aquarelblok	a : ka : r@lblOk
aquarellen	a : ka : rEI@n
aquarelletje	a : ka : rEI@tj@
aquarelletjes	a : ka : rEI@tj@s
aquarelpapier	a : ka : r@lpa : pir
aquarelschilder	a : ka : r@lsxIld@r
aquarelvelletjes	a : ka : r@lvEI@tj@s
aquarelverf	a : ka : r@lvErf
atelier	a : t@lir

orthographic	phonetic
beetje	be : tj@
begin	b@gIn
beginnen	b@gIn@n
behoefte	b@huft@
belangrijk	b@lANrEI
benen	be : n@n
bent	bEnt
bescherm laagje	bEsx@rmla : xj@
bestaan	b@sta : n
beter	be : t@r
bewerkelijk	b@wErk@l@k
bij	bEI
bijgepakt	bEIg@pAkt
bijna	bEI na :
bijvoorbeeld	bEIvo : rbe : lt
blauw	blAU
blijft	blEI ft
blijven	blEIv@n
blok	blOk
bobbelen	bOb@l@n
boeken	buk@n
boven	bo : v@n
bredere	bre : d@r@
broekzak	bruksAk
buiten	bUI t@n
cadmiumgeel	kAtmijUmge : l
cadmiumrood	kAtmijUmro : t
collectie	kOI Eksi
cursus	kUrzUs
daar	da : r
daarmee	da : rme :
daarom	da : rOm
dan	dAn

orthographic	phonetic
<i>dat</i>	<i>dAt</i>
<i>de</i>	<i>d@</i>
<i>deksel</i>	<i>dEks@l</i>
<i>denk</i>	<i>dENk</i>
<i>details</i>	<i>de : tAjls</i>
<i>deze</i>	<i>de : z@</i>
<i>dicht</i>	<i>dIxt</i>
<i>die</i>	<i>di</i>
<i>dik</i>	<i>dIk</i>
<i>dingen</i>	<i>dIN@n</i>
<i>dit</i>	<i>dIt</i>
<i>doe</i>	<i>du</i>
<i>doekje</i>	<i>dukj@</i>
<i>doen</i>	<i>dun</i>
<i>doet</i>	<i>dut</i>
<i>doorheen</i>	<i>do : rhe : n</i>
<i>doos</i>	<i>do : s</i>
<i>doosjes</i>	<i>do : sj@s</i>
<i>drie</i>	<i>dri</i>
<i>droog</i>	<i>dro : x</i>
<i>duidelijk</i>	<i>dUIId@l@k</i>
<i>dun</i>	<i>dUn</i>
<i>dus</i>	<i>dUs</i>
<i>duur</i>	<i>dyr</i>
<i>echt</i>	<i>Ext</i>
<i>echte</i>	<i>Ext@</i>
<i>een</i>	<i>@n</i>
<i>eens</i>	<i>e : ns</i>
<i>eentje</i>	<i>e : ntj@</i>
<i>eenvoudig</i>	<i>e : nvAUdIx</i>
<i>eerst</i>	<i>e : rst</i>
<i>eerste</i>	<i>e : rst@</i>
<i>eigenlijk</i>	<i>Elg@l@k</i>

orthographic	phonetic
<i>eindeloos</i>	<i>EInd@lo : s</i>
<i>elkaar</i>	<i>Elka : r</i>
<i>en</i>	<i>En</i>
<i>enorme</i>	<i>e : nOrm@</i>
<i>er</i>	<i>Er</i>
<i>erbij</i>	<i>ErbEI</i>
<i>erg</i>	<i>Erx</i>
<i>eronder</i>	<i>e : rOnd@r</i>
<i>even</i>	<i>e : v@n</i>
<i>experimenteer</i>	<i>Ekspe : rimEnte : r</i>
<i>extra</i>	<i>Ekstra :</i>
<i>fijn</i>	<i>fElN</i>
<i>fijne</i>	<i>fElN@</i>
<i>flink</i>	<i>fIINK</i>
<i>flinke</i>	<i>fIINK@</i>
<i>ga</i>	<i>xa :</i>
<i>gaan</i>	<i>xa : n</i>
<i>gaat</i>	<i>xa : t</i>
<i>geel</i>	<i>xe : l</i>
<i>geen</i>	<i>xe : n</i>
<i>gekleurd</i>	<i>x@klEUrt</i>
<i>gelijmd</i>	<i>x@lElmt</i>
<i>gemaakt</i>	<i>x@ma : kt</i>
<i>geperst</i>	<i>xe : p@rst</i>
<i>geschikt</i>	<i>x@sxIkt</i>
<i>geschilderd</i>	<i>x@sxIld@rt</i>
<i>gestructureerde</i>	<i>x@strUktyre : rd@</i>
<i>gewone</i>	<i>x@wo : n@</i>
<i>gewoon</i>	<i>x@wo : n</i>
<i>geworden</i>	<i>x@wOrd@n</i>
<i>gezien</i>	<i>x@zin</i>
<i>glad</i>	<i>xlAt</i>
<i>gladde</i>	<i>xlAd@</i>

orthographic	phonetic
<i>goed</i>	<i>xut</i>
<i>goede</i>	<i>xud@</i>
<i>goedkoop</i>	<i>xutko : p</i>
<i>grof</i>	<i>xrOf</i>
<i>groot</i>	<i>xro : t</i>
<i>grote</i>	<i>xro : t@</i>
<i>haar</i>	<i>ha : r</i>
<i>handel</i>	<i>hAnd@l</i>
<i>heb</i>	<i>hEp</i>
<i>hebben</i>	<i>hEb@n</i>
<i>hebt</i>	<i>hEpt</i>
<i>heeft</i>	<i>he : ft</i>
<i>heel</i>	<i>he : l</i>
<i>heerlijk</i>	<i>he : rl@k</i>
<i>heet</i>	<i>he : t</i>
<i>hele</i>	<i>he : l@</i>
<i>helemaal</i>	<i>he : l@ma : l</i>
<i>hem</i>	<i>hEm</i>
<i>het</i>	<i>@t</i>
<i>hier</i>	<i>hir</i>
<i>hij</i>	<i>hEI</i>
<i>hilversum</i>	<i>hIlv@rsUm</i>
<i>hoe</i>	<i>hu</i>
<i>hoeveel</i>	<i>huve : l</i>
<i>hoge</i>	<i>ho : g@</i>
<i>hoog</i>	<i>ho : x</i>
<i>hun</i>	<i>hUn</i>
<i>ie</i>	<i>i</i>
<i>iedereen</i>	<i>id@re : n</i>
<i>ik</i>	<i>Ik</i>
<i>in</i>	<i>In</i>
<i>is</i>	<i>Is</i>
<i>je</i>	<i>j@</i>

orthographic	phonetic
<i>kan</i>	<i>kAn</i>
<i>kant</i>	<i>kAnt</i>
<i>kapot</i>	<i>ka : pOt</i>
<i>keer</i>	<i>ke : r</i>
<i>keramiek</i>	<i>ke : ra : mik</i>
<i>kijk</i>	<i>kEIk</i>
<i>klaar</i>	<i>kla : r</i>
<i>klein</i>	<i>klEIIn</i>
<i>kleine</i>	<i>klEIIn@</i>
<i>kleuren</i>	<i>klEUr@n</i>
<i>knijp</i>	<i>knEIp</i>
<i>komen</i>	<i>ko : m@n</i>
<i>komende</i>	<i>ko : m@nd@</i>
<i>koop</i>	<i>ko : p</i>
<i>kopen</i>	<i>ko : p@n</i>
<i>kraan</i>	<i>kra : n</i>
<i>krijgt</i>	<i>krEIxt</i>
<i>kruk</i>	<i>krUk</i>
<i>kun</i>	<i>kUn</i>
<i>kunnen</i>	<i>kUn@n</i>
<i>kunt</i>	<i>kUnt</i>
<i>kwast</i>	<i>kwAst</i>
<i>kwasten</i>	<i>kwAst@n</i>
<i>laat</i>	<i>la : t</i>
<i>laatste</i>	<i>la : tst@</i>
<i>landschap</i>	<i>lAntsxAp</i>
<i>landschappen</i>	<i>lAntsxAp@n</i>
<i>lang</i>	<i>lAN</i>
<i>last</i>	<i>lAst</i>
<i>laten</i>	<i>la : t@n</i>
<i>leggen</i>	<i>lEg@n</i>
<i>lekker</i>	<i>lEk@r</i>
<i>lekkere</i>	<i>lEk@r@</i>

orthographic	phonetic
<i>les</i>	<i>lEs</i>
<i>lessen</i>	<i>lEs@n</i>
<i>leuk</i>	<i>lEUk</i>
<i>leuke</i>	<i>lEUk@</i>
<i>liefst</i>	<i>lifst</i>
<i>liever</i>	<i>liv@r</i>
<i>liggen</i>	<i>lIg@n</i>
<i>ligt</i>	<i>lIxt</i>
<i>los</i>	<i>lOs</i>
<i>lost</i>	<i>lOst</i>
<i>maakt</i>	<i>ma : kt</i>
<i>maar</i>	<i>ma : r</i>
<i>maken</i>	<i>ma : k@n</i>
<i>makkelijk</i>	<i>mAk@l@k</i>
<i>manier</i>	<i>ma : nir</i>
<i>marterhaar</i>	<i>mArt@rha : r</i>
<i>marters</i>	<i>mArt@rs</i>
<i>mat</i>	<i>mAt</i>
<i>maten</i>	<i>ma : t@n</i>
<i>matte</i>	<i>mAt@</i>
<i>max</i>	<i>mAks</i>
<i>mee</i>	<i>me :</i>
<i>meerdere</i>	<i>me ; rd@r@</i>
<i>meng</i>	<i>mEN</i>
<i>mensen</i>	<i>mEns@n</i>
<i>met</i>	<i>mEt</i>
<i>metaal</i>	<i>me : ta : l</i>
<i>mijn</i>	<i>mElIn</i>
<i>moe</i>	<i>mu</i>
<i>moet</i>	<i>mut</i>
<i>moeten</i>	<i>mut@n</i>
<i>mogelijk</i>	<i>mo : g@l@k</i>
<i>mooi</i>	<i>mo : j</i>

orthographic	phonetic
<i>mooie</i>	<i>mo : j@</i>
<i>n</i>	<i>n</i>
<i>naar</i>	<i>na : r</i>
<i>nadrukkelijke</i>	<i>na : drUk@l@k@</i>
<i>namelijk</i>	<i>na : m@l@k</i>
<i>napje</i>	<i>na : pj@</i>
<i>napjes</i>	<i>na : pj@s</i>
<i>nappen</i>	<i>na : p@n</i>
<i>nat</i>	<i>nAt</i>
<i>neem</i>	<i>ne : m</i>
<i>neemt</i>	<i>ne : mt</i>
<i>nemen</i>	<i>ne : m@n</i>
<i>net</i>	<i>nEt</i>
<i>niet</i>	<i>nit</i>
<i>nieuwe</i>	<i>niw@</i>
<i>niks</i>	<i>nIks</i>
<i>nodig</i>	<i>no : dIx</i>
<i>nog</i>	<i>nOx</i>
<i>of</i>	<i>Of</i>
<i>officieel</i>	<i>Ofisje : l</i>
<i>om</i>	<i>Om</i>
<i>onderwerpen</i>	<i>Ond@rwErp@n</i>
<i>ontzettend</i>	<i>OntsEt@nt</i>
<i>ook</i>	<i>o : k</i>
<i>op</i>	<i>Op</i>
<i>opgaan</i>	<i>Opxa : n</i>
<i>opgedroogd</i>	<i>Opx@dro : xt</i>
<i>opgedroogde</i>	<i>Opx@dro : gd@</i>
<i>opgespannen</i>	<i>Opx@spAn@n</i>
<i>oppervlak</i>	<i>Op@rvlAk</i>
<i>opzetten</i>	<i>OpsEt@n</i>
<i>oude</i>	<i>AUd@</i>
<i>overheen</i>	<i>o : v@rhe : n</i>

orthographic	phonetic
<i>paar</i>	<i>pa : r</i>
<i>pakken</i>	<i>pAk@n</i>
<i>pakt</i>	<i>pAkt</i>
<i>palet</i>	<i>pa : lEt</i>
<i>paletjes</i>	<i>pa : lEtj@s</i>
<i>papier</i>	<i>pa : pir</i>
<i>papieren</i>	<i>pa : pir@n</i>
<i>plakband</i>	<i>plAgbAnt</i>
<i>plank</i>	<i>plANk</i>
<i>plankje</i>	<i>plANkj@</i>
<i>porselein</i>	<i>pOrs@lEIn</i>
<i>pot</i>	<i>pOt</i>
<i>prachtig</i>	<i>prAxtix</i>
<i>prachtige</i>	<i>prAxsig@</i>
<i>prettig</i>	<i>prEtix</i>
<i>prima</i>	<i>prima :</i>
<i>probeer</i>	<i>pro : be : r</i>
<i>puntje</i>	<i>pUntj@</i>
<i>reservoir</i>	<i>r@zErvo : r</i>
<i>rondom</i>	<i>rOndOm</i>
<i>rood</i>	<i>ro : t</i>
<i>rug</i>	<i>rUx</i>
<i>samen</i>	<i>sa : m@n</i>
<i>schilderen</i>	<i>sxIld@r@n</i>
<i>schoon</i>	<i>sxo : n</i>
<i>schort</i>	<i>sxOrt</i>
<i>snel</i>	<i>snEl</i>
<i>snij</i>	<i>snEI</i>
<i>soorten</i>	<i>so : rt@n</i>
<i>spalters</i>	<i>spAlt@rs</i>
<i>spant</i>	<i>spAnt</i>
<i>speciaalzaak</i>	<i>spe : sja : lza : k</i>
<i>sponsje</i>	<i>spOnsj@</i>

orthographic	phonetic
<i>spullen</i>	<i>spUl@n</i>
<i>staand</i>	<i>sta : nt</i>
<i>staart</i>	<i>sta : rt</i>
<i>stem</i>	<i>stEm</i>
<i>stilleven</i>	<i>stIl@v@n</i>
<i>stillevens</i>	<i>stIl@v@ns</i>
<i>strak</i>	<i>strAk</i>
<i>structuur</i>	<i>strUktyr</i>
<i>tafel</i>	<i>ta : f@l</i>
<i>tape</i>	<i>ta : p@</i>
<i>te</i>	<i>t@</i>
<i>tip</i>	<i>tIp</i>
<i>tube</i>	<i>tyb@</i>
<i>tubes</i>	<i>tyb@s</i>
<i>twee</i>	<i>twe :</i>
<i>u</i>	<i>y</i>
<i>uit</i>	<i>UIt</i>
<i>uitglijdt</i>	<i>UItxlEIIt</i>
<i>uitproberen</i>	<i>UItpro : be : r@n</i>
<i>ultramarijn</i>	<i>Ultra : ma : rEIn</i>
<i>uw</i>	<i>yw</i>
<i>vakjes</i>	<i>vAkj@s</i>
<i>vakken</i>	<i>vAk@n</i>
<i>van</i>	<i>vAn</i>
<i>vanuit</i>	<i>va : nUIIt</i>
<i>vast</i>	<i>vAst</i>
<i>veel</i>	<i>ve : l</i>
<i>velletje</i>	<i>vEl@tj@</i>
<i>verdeelt</i>	<i>v@rde : lt</i>
<i>verf</i>	<i>vErF</i>
<i>verkocht</i>	<i>v@rkOxt</i>
<i>verschillende</i>	<i>v@rsxIl@nd@</i>
<i>versturen</i>	<i>v@rstyr@n</i>

orthographic	phonetic
<i>vervangen</i>	<i>v@rvAN@n</i>
<i>vervolgens</i>	<i>v@rvOlG@ns</i>
<i>vindt</i>	<i>vInt</i>
<i>vlak</i>	<i>vlAk</i>
<i>volgende</i>	<i>vOlG@nd@</i>
<i>voor</i>	<i>vo : r</i>
<i>voorkeur</i>	<i>vo : rkEUR</i>
<i>voorwerpje</i>	<i>vo : rwErpj@</i>
<i>vrij</i>	<i>vrEI</i>
<i>waar</i>	<i>wa : r</i>
<i>want</i>	<i>wAnt</i>
<i>wassen</i>	<i>wAs@n</i>
<i>wast</i>	<i>wAst</i>
<i>wat</i>	<i>wAt</i>
<i>water</i>	<i>wa : t@r</i>
<i>waterverf</i>	<i>wa : t@rvErF</i>
<i>we</i>	<i>w@</i>
<i>week</i>	<i>we : k</i>
<i>weer</i>	<i>we : r</i>
<i>weet</i>	<i>we : t</i>
<i>weinig</i>	<i>wElInIx</i>
<i>wel</i>	<i>wEl</i>
<i>werk</i>	<i>wErk</i>
<i>werken</i>	<i>wErk@n</i>
<i>werkt</i>	<i>wErkt</i>
<i>wij</i>	<i>wEI</i>
<i>wolkenlucht</i>	<i>wOlK@nlUxt</i>
<i>worden</i>	<i>wOrd@n</i>
<i>wordt</i>	<i>wOrt</i>
<i>ze</i>	<i>z@</i>
<i>zelf</i>	<i>zElf</i>
<i>zelfs</i>	<i>zElfs</i>
<i>zich</i>	<i>zIx</i>

orthographic	phonetic
<i>zichweer</i>	<i>zIxwe : r</i>
<i>zien</i>	<i>zin</i>
<i>ziet</i>	<i>zit</i>
<i>zijn</i>	<i>zElIn</i>
<i>zinnige</i>	<i>zInIg@</i>
<i>zit</i>	<i>zIt</i>
<i>zitten</i>	<i>zIt@n</i>
<i>zo</i>	<i>zo :</i>
<i>zoals</i>	<i>zo : wAls</i>
<i>zodat</i>	<i>zo : dAt</i>
<i>zorg</i>	<i>zOrx</i>

C WORD FREQUENCIES

This appendix contains the word frequencies of the television program "Iedereen kan aquarelschilderen". The word frequencies are generated with the Statistical Language Modeling Toolkit.

word	count
aan	4
absorbeert	1
actuele	1
advies	2
adviseren	2
afstand	2
afvegen	1
al	2
allebei	1
allereerst	2
allerlei	1
als	12
altijd	1
ander	1
anders	1
ansichtkaartje	1
aquarel	8
aquarelblok	1
aquarellen	1
aquarelletje	1
aquarelletjes	1
aquarelpapier	1
aquarelschilder	1
aquarelvelletjes	1
aquarelverf	4
atelier	1
beetje	1
begin	4
beginnen	1
behoefte	1
belangrijk	1
benen	1
bent	2
bescherm laagje	1
bestaan	1
beter	1
bewerkelijk	1
bij	1
bijgepakt	1
bijna	1
bijvoorbeeld	1

word	count
blauw	2
blijft	2
blijven	1
blok	1
bobbelen	1
boeken	1
boven	1
bredere	1
broekzak	1
buiten	1
cadmiumgeel	1
cadmiumrood	1
collectie	1
cursus	1
daar	4
daarmee	1
daarom	1
dan	11
dat	16
de	28
deksel	1
denk	1
details	2
deze	5
dicht	1
die	3
dik	1
dingen	1
dit	6
doe	2
doekje	1
doen	4
doet	1
doorheen	1
doos	1
doosjes	1
drie	3
droog	1
duidelijk	1
dun	3
dus	1

word	count
duur	1
echt	1
echte	1
een	50
eens	4
eentje	2
eenvoudig	1
eerst	2
eerste	1
eigenlijk	2
eindeloos	1
elkaar	1
en	34
enorme	1
er	20
erbij	1
erg	2
eronder	1
even	2
experimenteer	1
extra	1
fijn	1
fijne	1
flink	1
flinke	1
ga	1
gaan	2
gaat	4
geel	2
geen	1
gekleurd	1
gelijmd	1
gemaakt	1
geperst	1
geschikt	1
geschilderd	1
gestructureerde	1
gewone	2
gewoon	3
geworden	1
gezien	1

word	count
glad	4
gladde	1
goed	2
goede	4
goedkoop	1
grof	1
groot	3
grote	5
haar	1
handel	1
heb	15
hebben	5
hebt	2
heeft	2
heel	5
heerlijk	1
heet	1
hele	3
helemaal	1
hem	4
het	31
hier	11
hij	1
hilversum	1
hoe	3
hoeveel	1
hoge	1
hoog	1
hun	1
ie	6
iedereen	1
ik	14
in	15
is	30
je	51
kan	3
kant	2
kapot	1
keer	2
keramiek	1
kijk	2

word	count
klaar	3
klein	3
kleine	2
kleuren	4
knijp	2
komen	1
komende	1
koop	4
kopen	4
kraan	1
krijgt	2
kruk	1
kun	9
kunnen	1
kunt	6
kwast	5
kwasten	5
laat	3
laatste	1
landschap	2
landschappen	1
lang	1
last	1
laten	1
leggen	1
lekker	2
lekkere	2
les	1
lessen	1
leuk	2
leuke	1
liefst	1
liever	2
liggen	2
ligt	1
los	2
lost	1
maakt	2
maar	8
maken	7
makkelijk	3

word	count
manier	1
marterhaar	1
marters	1
mat	1
maten	1
matte	2
max	1
mee	7
meerdere	1
meng	2
mensen	3
met	22
metaal	1
mijn	4
moe	1
moet	2
moeten	2
mogelijk	1
mooi	4
mooie	2
n	1
naar	1
nadrukkelijke	1
namelijk	2
napje	1
napjes	4
nappen	1
nat	3
neem	3
neemt	1
nemen	2
net	1
niet	6
nieuwe	1
niks	1
nodig	8
nog	5
of	4
officieel	1
om	9
onderwerpen	3

word	count
ontzettend	1
ook	8
op	11
opgaan	1
opgedroogd	1
opgedroogde	1
opgespannen	1
oppervlak	1
opzetten	1
oude	1
overheen	2
paar	3
pakken	1
pakt	1
palet	3
paletjes	1
papier	18
papieren	2
plakband	1
plank	1
plankje	1
porselein	1
pot	1
prachtig	2
prachtige	1
prettig	2
prima	2
probeer	4
puntje	3
reservoir	2
rondom	2
rood	2
rug	1
samen	1
schilderen	4
schoon	1
schort	1
snel	2
snij	1

word	count
soorten	4
spalters	1
spant	1
speciaalzaak	1
sponsje	1
spullen	1
staand	1
staart	1
stem	1
stilleven	1
stillevens	1
strak	2
structuur	2
tafel	5
tape	1
te	12
tip	1
tube	1
tubes	3
twee	1
u	2
uit	6
uitglijdt	1
uitproberen	1
ultramarijn	1
uw	1
vakjes	1
vakken	2
van	16
vanuit	1
vast	2
veel	8
velletje	1
verdeelt	1
verf	11
verkocht	1
verschillende	3
versturen	1
vervangen	1
vervolgens	1
vindt	1

word	count
vlak	1
volgende	1
voor	10
voorkeur	1
voorwerpje	1
vrij	1
waar	3
want	2
wassen	1
wast	1
wat	14
water	9
watervarf	1
we	6
week	1
weer	3
weet	1
weinig	2
wel	4
werk	2
werken	3
werkt	6
wij	1
wolkenlucht	1
worden	2
wordt	3
ze	3
zelf	2
zelfs	2
zich	1
zichweer	1
zien	4
ziet	3
zijn	12
zinnige	1
zit	2
zitten	2
zo	9
zoals	3
zodat	1
zorg	1

D OUTPUT OF MATCH

This appendix lists non-zero score candidate ASR segments for the caption "Dit is de eerste les van de cursus: Iedereen kan aquarellen." from the television program "Iedereen kan aquarelschilderen". The candidates are created by sliding a window with a size equal to the caption length.

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["hij","is","moeten","maken","van","een","een","is","eerste","les","van"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [86720,146080,190240,215360,262880,306560,417440,503680,508000,513760,519040]
new offset : 147840

score : 17.256382
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["een","een","is","eerste","les","van","het","is","is","iedereen","kan"]
matches : [0.0,0.0,0.0,7.1395516,7.1395516,2.9772797,0.0,0.0,0.0,0.0,0.0]
inframes : [306560,417440,503680,508000,513760,519040,521280,523840,527520,532640,541600]
new offset : 521280

score : 21.713547
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["een","is","is","eerste","les","van","het","is","is","iedereen","kan","aquarellen"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,7.1395516,5.5339785,7.1395516]
inframes : [417440,503680,508000,513760,519040,521280,523840,527520,532640,541600,545760]
new offset : 545760

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["het","is","is","iedereen","kan","aquarellen","schilderen","erbij","gaan","deze","les"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [521280,523840,527520,532640,541600,545760,553920,582720,585600,588480,593120]
new offset : 527520

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["is","is","iedereen","kan","aquarellen","schilderen","erbij","gaan","deze","les","en"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [523840,527520,532640,541600,545760,553920,582720,585600,588480,593120,597600]
new offset : 532000

score : 9.165087
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["schilderen","erbij","gaan","deze","les","en","de","komende","die","deze","samen"]
matches : [0.0,0.0,0.0,0.0,7.1395516,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes : [553920,582720,585600,588480,593120,597600,600960,602400,608000,612640,621280]
new offset : 602400

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["les","en","de","komende","die","deze","samen","en","aquarel","maken","zelfs"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [593120,597600,600960,602400,608000,612640,621280,631360,633760,640800,662240]
new offset : 602400

score : 2.9772797
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["aquarel","maken","zelfs","meerdere","maken","van","een","paar","hier","op","het"]
matches : [0.0,0.0,0.0,0.0,0.0,2.9772797,0.0,0.0,0.0,0.0,0.0]

inframes : [633760,640800,662240,665600,678880,684480,687040,689120,693440,696960,699200]
new offset : 687040

score : 5.002815
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["op","het","atelier","in","maken","van","de","buiten","doen","en","gaan"]
matches : [0.0,0.0,0.0,0.0,0.0,2.9772797,2.025535,0.0,0.0,0.0,0.0]
inframes : [696960,699200,700960,705600,708160,720480,723520,725280,731360,736480,739520]
new offset : 725280

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["maken","van","de","buiten","doen","en","gaan","landschap","schilderen","laatste","eens"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [708160,720480,723520,725280,731360,736480,739520,744480,753120,771200,776480]
new offset : 725280

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi :
["laatste","eens","en","landschappen","dat","zijn","de","onderwerpen","voor","de","aquarelschilder"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes : [771200,776480,784000,785760,796000,798240,802400,808480,818400,820800,822400]
new offset : 808480

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi :
["landschappen","dat","zijn","de","onderwerpen","voor","de","aquarelschilder","dat","is","ander"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes : [785760,796000,798240,802400,808480,818400,820800,822400,852480,854880,857280]
new offset : 822400

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dat","zijn","de","onderwerpen","voor","de","aquarelschilder","dat","is","ander","voor"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [796000,798240,802400,808480,818400,820800,822400,852480,854880,857280,865280]
new offset : 808480

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["onderwerpen","voor","de","aquarelschilder","dat","is","ander","voor","het","maken","van"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [808480,818400,820800,822400,852480,854880,857280,865280,867840,868800,874560]
new offset : 822400

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dat","is","ander","voor","het","maken","van","aquarel","allereerst","een","gewone"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [852480,854880,857280,865280,867840,868800,874560,878720,900960,909600,910560]
new offset : 857280

score : 2.9772797
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["is","ander","voor","het","maken","van","aquarel","allereerst","een","gewone","tafel"]
matches : [0.0,0.0,0.0,0.0,0.0,2.9772797,0.0,0.0,0.0,0.0,0.0]
inframes : [854880,857280,865280,867840,868800,874560,878720,900960,909600,910560,918080]
new offset : 878720

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["van","aquarel","allereerst","een","gewone","tafel","de","tafel","is","drie","maken"]
matches : [0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes : [874560,878720,900960,909600,910560,918080,930400,936640,941920,944800,948320]
new offset : 936640

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["gewone","tafel","de","tafel","is","drie","maken","deze","in","de","niet"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [910560,918080,930400,936640,941920,944800,948320,966560,969760,971840,979520]
new offset : 936640

score : 3.9259994
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["tafel","is","drie","maken","deze","in","de","niet","wordt","aquarelverf","is"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes : [936640,941920,944800,948320,966560,969760,971840,979520,982720,987040,998560]
new offset : 978400

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["deze","in","de","niet","wordt","aquarelverf","is","heel","schoon","in","kunt"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes : [966560,969760,971840,979520,982720,987040,998560,1000480,1004160,1013760,1015040]
new offset : 978400

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["aquarelverf","is","heel","schoon","in","kunt","het","zo","met","een","doekje"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[987040,998560,1000480,1004160,1013760,1015040,1018080,1019360,1022400,1024480,1025760]
new offset : 1000480

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dat","is","al","aan","dit","weer","wat","aquarelverf","is","ontzettend","makkelijk"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1097760,1112480,1119520,1122400,1124800,1127840,1132160,1140800,1152640,1156000,1162400]
new offset : 1119520

score : 4.5025

title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dit","weer","wat","aquarelverf","is","ontzettend","makkelijk","dat","is","waterverf","groot"]
matches : [4.5025,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1124800,1127840,1132160,1140800,1152640,1156000,1162400,1168800,1172480,1175360,1197280]
new offset : 1127840

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["aquarelverf","is","ontzettend","makkelijk","dat","is","waterverf","groot","ik","zelf","werkt"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1140800,1152640,1156000,1162400,1168800,1172480,1175360,1197280,1207040,1209440,1212960]
new offset : 1156000

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dat","is","waterverf","groot","ik","zelf","werkt","die","staand","aan","tafel"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1168800,1172480,1175360,1197280,1207040,1209440,1212960,1221120,1225280,1233120,1236800]
new offset : 1175360

score : 2.9772797
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["om","te","en","geen","last","van","rug","krijgt","maar","voor","is"]
matches : [0.0,0.0,0.0,0.0,0.0,2.9772797,0.0,0.0,0.0,0.0,0.0]
inframes :
[1347840,1351520,1353280,1359200,1362560,1366720,1370720,1374720,1402240,1404480,1407680]
new offset : 1370720

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["voor","is","laatste","afstand","als","je","staand","zijn","gewone","tafel","prima"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1404480,1407680,1413120,1418400,1424960,1427840,1428960,1435040,1438720,1445600,1452160]
new offset : 1413120

score : 7.1395516
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["goed","zien","wat","doet","mensen","die","moe","worden","iedereen","kunnen","zien"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,7.1395516,0.0,0.0]
inframes :
[1494080,1497600,1500320,1504960,1526080,1530720,1531840,1535040,1539520,1559200,1564320]
new offset : 1551520

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["iedereen","kunnen","zien","dan","al","elkaar","de","waar","en","daar","ook"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes :

[1539520,1559200,1564320,1574560,1577600,1580480,1586400,1587680,1606720,1608640,1611680]
new offset : 1587680

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["al","elkaar","de","waar","en","daar","ook","zitten","maar","dan","voor"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1577600,1580480,1586400,1587680,1606720,1608640,1611680,1613600,1623040,1631040,1634720]
new offset : 1587680

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["voor","dat","je","flink","hoog","boven","de","tafel","blijft","zitten","voor"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes :
[1634720,1638560,1641600,1657120,1661760,1666880,1672480,1674560,1680320,1684800,1704800]
new offset : 1674560

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["hoog","boven","de","tafel","blijft","zitten","voor","en","veel","dat","is"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1661760,1666880,1672480,1674560,1680320,1684800,1704800,1726560,1741760,1758400,1761760]
new offset : 1674560

score : 7.1395516
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["voor","en","veel","dat","is","namelijk","voor","het","maken","van","aquarellen"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,7.1395516]
inframes :
[1704800,1726560,1741760,1758400,1761760,1763680,1771680,1773920,1774880,1780640,1784320]
new offset : 1726560

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dat","is","namelijk","voor","het","maken","van","aquarellen","allereerst","een","hele"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1758400,1761760,1763680,1771680,1773920,1774880,1780640,1784320,1805920,1812640,1814720]
new offset : 1763680

score : 2.9772797
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["is","namelijk","voor","het","maken","van","aquarellen","allereerst","een","hele","water"]
matches : [0.0,0.0,0.0,0.0,0.0,2.9772797,0.0,0.0,0.0,0.0,0.0]
inframes :
[1761760,1763680,1771680,1773920,1774880,1780640,1784320,1805920,1812640,1814720,1818080]
new offset : 1782880

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]

candi : ["oude","anders","pot","schoon","water","uit","de","kraan","is","prima","om"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes :
[1854560,1865440,1871360,1885920,1890880,1896480,1899040,1900000,1905760,1908800,1913600]
new offset : 1900000

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["water","uit","de","kraan","is","prima","om","mee","te","werken","voor"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1890880,1896480,1899040,1900000,1905760,1908800,1913600,1916000,1918720,1920000,1945920]
new offset : 1900000

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["kraan","is","prima","om","mee","te","werken","voor","het","je","de"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1900000,1905760,1908800,1913600,1916000,1918720,1920000,1945920,1950560,1954400,1955520]
new offset : 1908320

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["mee","te","werken","voor","het","je","de","verf","aquarelverf","wordt","verkocht"]
matches : [0.0,0.0,0.0,0.0,0.0,0.0,2.025535,0.0,0.0,0.0,0.0]
inframes :
[1916000,1918720,1920000,1945920,1950560,1954400,1955520,1957760,1965120,1975200,1978080]
new offset : 1957760

score : 2.025535
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["het","je","de","verf","aquarelverf","wordt","verkocht","in","dit","is","op"]
matches : [0.0,0.0,2.025535,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1950560,1954400,1955520,1957760,1965120,1975200,1978080,1984800,1989440,1994720,2009120]
new offset : 1957760

score : 6.4029646
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["dit","is","op","in","nog","je","is","je","kunt","napje","daar"]
matches : [4.5025,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[1989440,1994720,2009120,2013120,2017440,2023840,2026720,2042560,2046240,2048960,2054400]
new offset : 2001120

score : 1.9004643
title : ["dit","is","de","eerste","les","van","de","cursus","iedereen","kan","aquarellen"]
candi : ["je","is","je","kunt","napje","daar","zit","opgedroogd","aan","wel","verf"]
matches : [0.0,1.9004643,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inframes :
[2023840,2026720,2042560,2046240,2048960,2054400,2056640,2072960,2082240,2085280,2089600]
new offset : 2032160


```
#!/bin/sh

while true;
do
    until ls ../upload | grep .info
    do
        sleep 1
        nice -n 19 date;echo "SETH is waiting for files"
    done

    echo "Start import"
    perl npobuilder.pl -import > import.log

    echo "Start train"
    perl npobuilder.pl -train > train.log

    echo "Start test"
    perl npobuilder.pl -test

    echo "Start align"
    perl npobuilder.pl -align > align.log

    echo "Start export"
    perl npobuilder.pl -export > export.log

    sleep 5
    echo "Start clean"
    perl npobuilder.pl -clean

    echo "Loop"
done
```

F NORMALIZED TRAINFILE

This appendix shows a text-file suitable for training a SONIC SLM and lexicon. The text-file contains concatenated and cleaned captions from the television program "NOVA".

Blad1

<s> lange tijd hing hem een gevangenisstraf van een vier jaar boven het hoofd in een braziliaanse cel </s>
<s> de nederlandse bioloog marc van roosmalen vermaard vanwege de ontdekking van een nieuwe apensoort </s>
<s> hij werd door de braziliaanse overheid verdacht van apendiefstal </s>
<s> onlangs kwam hij na een maandenlangeprocedure vrij </s>
<s> en voor het eerst sinds jaren is van roosmalen even in nederland </s>
<s> de man die veel heeft gedaan om hem vrij te krijgen stond hem op te wachten op schiphol </s>
<s> een verslag van marcel ouddeken en silvia pilger </s>
<s> muziek </s>
<s> eh </s>
<s> een een uur twee nul </s>
<s> ja expect landing </s>
<s> ja een een uur drie negen </s>
<s> de gepensioneerde braziliaanse advocaat gau lyra </s>
<s> heeft marc van roosmalen uit de braziliaanse cel gekregen </s>
<s> dat werk deed hij allemaal vanuit nederland </s>
<s> vandaag gaat hij na twee jaar inspanningen zijn client voor het eerst van zijn leven ontmoeten </s>
<s> ik weet hoe hij eruitziet ik weet hoe hij denkt hoe hij praat </s>
<s> ik weet alles van marc </s>
<s> alleen hebben wij nooit lijfelijk tegenover elkaar gestaan </s>
<s> en nu gaat dat gebeuren </s>
<s> ik moet wel zeggen ik vind het spannend </s>
<s> beetje beetje zenuwachtig </s>
<s> meneer lyra was begin jaren zeven nul slachtoffer van martelingen door het braziliaanse militaire regime </s>
<s> reden waarom hij zich nu zo heeft ingezet voor de bevrijding van marc </s>
<s> muziek </s>
<s> muziek </s>
<s> marc van roosmalen bioloog </s>
<s> of beter gezegd primatoloog apenexpert </s>
<s> bij zijn woning in brazilie verzorgde hij bedreigde apen </s>
<s> en dat mag niet zegt de braziliaanse rechter </s>
<s> van roosmalen is veroordeeld voor het stelen van apen </s>
<s> hij heeft een illegale apenopvang en daar tillen ze zwaar aan in brazilie </s>
<s> van roosmalen is tot een vier jaar gevangenisstraf veroordeeld </s>
<s> marc zegt behalve aan gau ook veel te danken te hebben </s>
<s> aan zijn vriendin vivian die heeft hij in brazilie ontmoet </s>
<s> in het portugees zeg je valeo valeo pena </s>
<s> het was echt de moeite waard </s>
<s> om door al die ellende heen te gaan </s>
<s> en mensen te leren kennen als vivian en gau en ricardo </s>
<s> en dat vind ik de mooie dingen van het leven </s>
<s> muziek </s>
<s> in zo n gevangenis om te overleven </s>
<s> omdat daar zo veel drugs verhandeld worden binnen de gevangenis </s>
<s> moet je dus overleven met misdadigers </s>

G SONIC CONFIGURATION FILE

This appendix displays the Sonic configuration file for the recognition of the television program "Boeken".

-batch_file ../models/boeken_09-11_0.bat

-output_file ../models/boeken_09-11_0.hyp
-log_file ../models/boeken_09-11.log
-log_dir ../models/
-expand_compound_words 0
-output_pronunciations 0
-align_word_hypothesis 1

-phone_config ../models/phoneset.txt
-dictionary ../models/boeken_09-11.lex.bin

-acoustic_mod ../models/am.2.bin
-langmod_file ../models/boeken_09-11.lm.bin
-filler_file ../models/filler.wlist
-filler_penalty -10.0

-sample_rate 16000.0
-feature_type PMVDR

-word_end_beam 100.0
-word_entry_beam 100.0
-state_beam 200.0
-lm_scale 30.0
-max_active_states 40000
-max_word_ends 1000
-word_trans_penalty -16.5
-state_dur_scale 2.0
-short_word_penalty -10.0
-lm_garbage_collect 1
-end_point_padding 125

H ALIGNED CAPTIONS

```
<?xml version="1.0" ?>
<titles>
  <title nr="0001">
    <caption>Dit is de eerste les van de cursus: Iedereen kan aquarellen.</caption>
    <hypothesis>voor is laatste afstand als je staand zijn gewone tafel prima</hypothesis>
    <tci>00:00:26:09</tci>
    <tco>00:00:30:64</tco>
    <score>21.713547</score>
  </title>
  <title nr="0002">
    <caption>Wij gaan de komende lessen samen een aquarel maken.</caption>
    <hypothesis>die dan plank op tafel leggen een paar boeken</hypothesis>
    <tci>00:00:37:35</tci>
    <tco>00:00:41:07</tco>
    <score>15.475101</score>
  </title>
  <title nr="0003">
    <caption>We maken er zelfs meerdere. Een paar hier op het atelier.</caption>
    <hypothesis>dan plank op tafel leggen een paar boeken eronder zodat wat</hypothesis>
    <tci>00:00:40:05</tci>
    <tco>00:00:44:60</tco>
    <score>20.70122</score>
  </title>
  <title nr="0004">
    <caption>En we gaan ook buiten een landschap schilderen.</caption>
    <hypothesis>en geen last van rug krijgt maar voor</hypothesis>
    <tci>00:00:45:03</tci>
    <tco>00:00:48:34</tco>
    <score>11.237807</score>
  </title>
  <title nr="0005">
    <caption>Want stillevens en landschappen zijn DE onderwerpen...</caption>
    <hypothesis>om te en geen last van rug</hypothesis>
    <tci>00:00:48:53</tci>
    <tco>00:00:51:43</tco>
    <score>8.971384</score>
  </title>
  <title nr="0006">
    <caption>voor de aquarelschilder.</caption>
    <hypothesis>voor het maken</hypothesis>
    <tci>00:00:51:15</tci>
    <tco>00:00:52:39</tco>
    <score>10.862018</score>
  </title>
  <title nr="0007">
    <caption>Wat hebben we nodig voor het maken van een aquarel?</caption>
    <hypothesis>ook zitten maar dan voor dat je flink hoog boven</hypothesis>
    <tci>00:00:51:40</tci>
    <tco>00:00:55:54</tco>
    <score>10.969206</score>
  </title>
  <title nr="0008">
    <caption>Allereerst een gewone tafel.</caption>
    <hypothesis>die staand aan tafel</hypothesis>
    <tci>00:00:56:31</tci>
    <tco>00:00:57:97</tco>
    <score>17.914375</score>
  </title>
  <title nr="0009">
    <caption>Aquarelverf is heel schoon.</caption>
    <hypothesis>tafel wat heel lang</hypothesis>
    <tci>00:01:01:69</tci>
    <tco>00:01:03:35</tco>
    <score>18.924644</score>
  </title>
  <title nr="0010">
    <caption>Je kunt het zo met een doekje weer afvegen.</caption>
    <hypothesis>het maken van aquarellen allereerst een hele water neem</hypothesis>
    <tci>00:01:03:36</tci>
    <tco>00:01:07:08</tco>
    <score>24.388552</score>
  </title>
  <title nr="0011">
    <caption>Ik heb wel een schort om, maar dat is eigenlijk niet nodig.</caption>
    <hypothesis>opgedroogd aan wel verf in en zijn eentje kwast overheen dan lost</hypothesis>
    <tci>00:01:05:80</tci>
    <tco>00:01:10:77</tco>
    <score>14.398286</score>
  </title>
</titles>
```

```

</title>
<title nr="0012">
  <altcaption>No alternative caption found!</altcaption>
  <caption>Je wast het er zo uit. Het is namelijk waterverf.</caption>
  <hypothesis>de verf aquarelverf wordt verkocht in dit is op in</hypothesis>
  <tci>00:01:10:77</tci>
  <tco>00:01:14:83</tco>
  <score>8.1</score>
</title>
<title nr="0013">
  <caption>Ik werk zelf het liefst staand aan tafel.</caption>
  <hypothesis>voor dat je flink hoog boven de tafel</hypothesis>
  <tci>00:01:14:83</tci>
  <tco>00:01:18:14</tco>
  <score>23.153461</score>
</title>
<title nr="0014">
  <caption>Als u heel lang bent, kunt een plank...</caption>
  <hypothesis>allereerst een hele water neem gewoon een lekkere</hypothesis>
  <tci>00:01:17:30</tci>
  <tco>00:01:20:61</tco>
  <score>11.915655</score>
</title>
<title nr="0015">
  <caption>op tafel leggen met wat boeken eronder.</caption>
  <hypothesis>de tafel blijft zitten voor en veel</hypothesis>
  <tci>00:01:20:90</tci>
  <tco>00:01:23:80</tco>
  <score>29.768751</score>
</title>
<title nr="0016">
  <caption>Dan krijgt u geen last van uw rug.</caption>
  <hypothesis>is namelijk voor het maken van aquarellen allereerst</hypothesis>
  <tci>00:01:24:24</tci>
  <tco>00:01:27:55</tco>
  <score>17.256382</score>
</title>
<title nr="0017">
  <caption>De afstand met een gewone tafel is prima.</caption>
  <hypothesis>dit is op in nog je is je</hypothesis>
  <tci>00:01:29:06</tci>
  <tco>00:01:32:37</tco>
  <score>10.905387</score>
</title>
<title nr="0018">
  <caption>Dan heb je een goede afstand en kun je goed zien wat je doet.</caption>
  <hypothesis>doosjes met de napjes de kant en klaar in is als deze hier als</hypothesis>
  <tci>00:01:31:52</tci>
  <tco>00:01:37:31</tco>
  <score>21.921608</score>
</title>
<title nr="0019">
  <caption>Mensen die moe worden in hun benen kunnen een hoge kruk pakken.</caption>
  <hypothesis>in tubes is in in napjes en gaan dat heeft probeer wat</hypothesis>
  <tci>00:01:35:38</tci>
  <tco>00:01:40:35</tco>
  <score>24.336792</score>
</title>
<title nr="0020">
  <caption>Zorg dan wel dat je flink hoog boven de tafel blijft zitten.</caption>
  <hypothesis>van napjes daar kun je op maar de extra kopen in handel</hypothesis>
  <tci>00:01:41:44</tci>
  <tco>00:01:46:41</tco>
  <score>45.839962</score>
</title>
<title nr="0021">
  <caption>Wat is er nodig voor het maken van een aquarel?</caption>
  <hypothesis>ik is van napjes daar kun je op maar de</hypothesis>
  <tci>00:01:48:86</tci>
  <tco>00:01:53:00</tco>
  <score>10.969206</score>
</title>
<title nr="0022">
  <caption>Allereerst water.</caption>
  <altcaption>INS allereerst INS INS water</altcaption>
  <hypothesis start="111.52">aquarellen allereerst een hele water</hypothesis>
  <tci>00:01:51:52</tci>
  <tco>00:01:52:52</tco>
  <score>10.014641</score>
</title>
<title nr="0023">
  <caption>Neem gewoon een lekkere oude pot.</caption>
  <hypothesis>gewoon een lekkere oude anders pot</hypothesis>
  <tci>00:01:54:73</tci>

```

<tco>00:01:57:21</tco>
 <score>25.216496</score>
 </title>
 <title nr="0024">
 <caption>Water uit de kraan is prima om mee te werken.</caption>
 <hypothesis>kant en klaar in is als deze hier als eentje</hypothesis>
 <tci>00:01:58:18</tci>
 <tco>00:02:02:32</tco>
 <score>40.683357</score>
 </title>
 <title nr="0025">
 <caption>Vervolgens verf, aquarelverf wordt verkocht in tubes.</caption>
 <hypothesis>als dan is ik niet voor tubes</hypothesis>
 <tci>00:02:02:22</tci>
 <tco>00:02:05:12</tco>
 <score>24.437841</score>
 </title>
 <title nr="0026">
 <caption>Of in napjes.</caption>
 <hypothesis>verf in en</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>3.081794</score>
 </title>
 <title nr="0027">
 <caption>Hier heb ik een napje met opgedroogde aquarelverf.</caption>
 <hypothesis>zijn ook helemaal in doosjes met de napjes</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>7.1395516</score>
 </title>
 <title nr="0028">
 <caption>Als je er met je kwast overheen gaat lost het op.</caption>
 <hypothesis>nappen die zijn hele lekker als zich gaat met grote kwasten</hypothesis>
 <tci>00:02:10:60</tci>
 <tco>00:02:15:15</tco>
 <score>21.61893</score>
 </title>
 <title nr="0029">
 <caption>En vanuit de tube knijp je het op je palet.</caption>
 <hypothesis>grote kwasten werkt dan heb je gaan ook groot oppervlak</hypothesis>
 <tci>00:02:13:22</tci>
 <tco>00:02:17:36</tco>
 <score>17.08289</score>
 </title>
 <title nr="0030">
 <caption>Je hebt maar weinig nodig en daar doe je water bij.</caption>
 <hypothesis>je de verf uit of de in napjes en hier in</hypothesis>
 <tci>00:02:17:04</tci>
 <tco>00:02:21:59</tco>
 <score>18.82493</score>
 </title>
 <title nr="0031">
 <caption>De tubes zijn dus vrij klein.</caption>
 <hypothesis>in handel zijn ook helemaal in</hypothesis>
 <tci>00:02:20:59</tci>
 <tco>00:02:23:07</tco>
 <score>12.673531</score>
 </title>
 <title nr="0032">
 <caption>Mijn advies is om van allebei een paar te kopen.</caption>
 <hypothesis>zijn dat is een palet hier in die kleine vakjes</hypothesis>
 <tci>00:02:22:70</tci>
 <tco>00:02:26:84</tco>
 <score>27.14046</score>
 </title>
 <title nr="0033">
 <caption>Koop bijvoorbeeld rood,geel en blauw in tubes en in napjes.</caption>
 <hypothesis>of de in napjes en hier in de grote vakken meng</hypothesis>
 <tci>00:02:28:50</tci>
 <tco>00:02:33:05</tco>
 <score>38.892395</score>
 </title>
 <title nr="0034">
 <caption>En probeer dan zelf wat je prettig vindt om mee te werken.</caption>
 <hypothesis>zo weer nieuwe ik heb je een prachtig een collectie kwasten goede</hypothesis>
 <tci>00:02:33:39</tci>
 <tco>00:02:38:36</tco>
 <score>34.42305</score>
 </title>
 <title nr="0035">
 <caption>Als je weet waar je voorkeur ligt kun je wat kleuren extra kopen.</caption>
 <hypothesis>van marters maken hele aquarel mee maken zijn voor wat haar aan dit</hypothesis>
 <tci>00:02:40:00</tci>

<tco>00:02:45:38</tco>
 <score>12.2480755</score>
 </title>
 <title nr="0036">
 <caption>Er zijn ook mooie doosjes met de napjes er kant en klaar in.</caption>
 <hypothesis>schilderen zijn dat je kwast in handel spalters is heel geschikt om groot</hypothesis>
 <tci>00:02:44:01</tci>
 <tco>00:02:49:39</tco>
 <score>31.108013</score>
 </title>
 <title nr="0037">
 <caption>Zoals deze hier.</caption>
 <hypothesis>als deze hier</hypothesis>
 <tci>00:02:47:42</tci>
 <tco>00:02:48:66</tco>
 <score>8.350096</score>
 </title>
 <title nr="0038">
 <caption>Als er eentje op is , kun je hem gewoon vervangen.</caption>
 <hypothesis>wat haar aan dit is het deze maar voor werkt</hypothesis>
 <tci>00:02:47:96</tci>
 <tco>00:02:52:10</tco>
 <score>15.489097</score>
 </title>
 <title nr="0039">
 <caption>Paletjes in de deksel.</caption>
 <hypothesis>hier in die kleine</hypothesis>
 <tci>00:02:52:56</tci>
 <tco>00:02:54:22</tco>
 <score>17.360897</score>
 </title>
 <title nr="0040">
 <caption>En zo gaat-ie weer dicht.</caption>
 <hypothesis>en hier in de grote vakken</hypothesis>
 <tci>00:02:54:73</tci>
 <tco>00:02:57:21</tco>
 <score>27.837337</score>
 </title>
 <title nr="0041">
 <caption>Hier heb ik nog een grote doos met hele grote nappen.</caption>
 <hypothesis>een prachtig mooi de de eentje en daar hele kleine details</hypothesis>
 <tci>00:02:55:50</tci>
 <tco>00:03:00:05</tco>
 <score>19.139217</score>
 </title>
 <title nr="0042">
 <caption>Die zijn lekker als je met grote kwasten werkt.</caption>
 <hypothesis>voor werkt en als ie nat maakt krijgt ie</hypothesis>
 <tci>00:03:03:11</tci>
 <tco>00:03:06:83</tco>
 <score>16.502167</score>
 </title>
 <title nr="0043">
 <caption>Dan heb je snel een groot oppervlak gekleurd met verf.</caption>
 <hypothesis>spalters is heel geschikt om groot vlak in te wassen</hypothesis>
 <tci>00:03:05:45</tci>
 <tco>00:03:09:59</tco>
 <score>33.325684</score>
 </title>
 <title nr="0044">
 <altcaption>No alternative caption found!</altcaption>
 <caption>Dan heb je een palet nodig.</caption>
 <hypothesis>je een prachtig een collectie kwasten</hypothesis>
 <tci>00:03:09:59</tci>
 <tco>00:03:15:33</tco>
 <score>8.1</score>
 </title>
 <title nr="0045">
 <caption>Hier in de kleine vakjes knijp je de verf uit of doe je de napjes.</caption>
 <hypothesis>aquarelpapier in de handel duur papier goedkoop papier dik papier dun dat weer glad papier</hypothesis>
 <tci>00:03:15:33</tci>
 <tco>00:03:21:54</tco>
 <score>33.945004</score>
 </title>
 <title nr="0046">
 <caption>En in de grote vakken meng je de verf met water.</caption>
 <hypothesis>en eens goede kwast hele aquarel schilderen om ik er even</hypothesis>
 <tci>00:03:20:52</tci>
 <tco>00:03:25:07</tco>
 <score>30.858643</score>
 </title>
 <title nr="0047">
 <caption>Officieel moeten ze van keramiek , porselein of metaal zijn.</caption>
 <hypothesis>om groot vlak in te wassen ook laatste zijn</hypothesis>

<tc1>00:03:24:31</tc1>
 <tco>00:03:28:03</tco>
 <score>51.74468</score>
 </title>
 <title nr="0048">
 <caption>Maar dit werkt erg goed en is makkelijk mee te nemen.</caption>
 <hypothesis>wat wat hier mijn maakt dat is het hele ook belangrijk</hypothesis>
 <tc1>00:03:27:73</tc1>
 <tco>00:03:32:28</tco>
 <score>21.270594</score>
 </title>
 <title nr="0049">
 <caption>En als-ie kapot gaat koop je zo weer een nieuwe.</caption>
 <hypothesis>is het hele ook belangrijk voor het maken van een aquarel</hypothesis>
 <tc1>00:03:31:61</tc1>
 <tco>00:03:36:16</tco>
 <score>37.21863</score>
 </title>
 <title nr="0050">
 <caption>Ik heb een prachtige collectie kwasten.</caption>
 <hypothesis>kwast met een grote deze waar</hypothesis>
 <tc1>00:03:36:05</tc1>
 <tco>00:03:38:53</tco>
 <score>11.915655</score>
 </title>
 <title nr="0051">
 <caption>Goede kwasten zijn van marterhaar gemaakt.</caption>
 <hypothesis>ook laatste zijn meerdere nog snel</hypothesis>
 <tc1>00:03:39:02</tc1>
 <tco>00:03:41:50</tco>
 <score>30.578415</score>
 </title>
 <title nr="0052">
 <caption>Zoals deze hier.</caption>
 <hypothesis>het deze maar</hypothesis>
 <tc1>00:03:41:71</tc1>
 <tco>00:03:42:95</tco>
 <score>8.350096</score>
 </title>
 <title nr="0053">
 <caption>Van het puntje van de staart van marters.</caption>
 <hypothesis>hele ook belangrijk voor het maken van een</hypothesis>
 <tc1>00:03:42:16</tc1>
 <tco>00:03:45:47</tco>
 <score>20.233662</score>
 </title>
 <title nr="0054">
 <caption>Daar kun je een hele aquarel mee maken.</caption>
 <hypothesis>het maken van een aquarel hier drie verschillende</hypothesis>
 <tc1>00:04:23:96</tc1>
 <tco>00:04:27:27</tco>
 <score>9.600067</score>
 </title>
 <title nr="0055">
 <caption>Er zit veel haar aan. Dit is het reservoir voor de verf.</caption>
 <hypothesis>papier wel die ziet aan mijn kwast ziet ander de structuur van</hypothesis>
 <tc1>00:00:00:00</tc1>
 <tco>00:00:00:00</tco>
 <score>5.474456</score>
 </title>
 <title nr="0056">
 <caption>En als je hem nat maakt, krijgt-ie een prachtig puntje.</caption>
 <hypothesis>en heeft niet echt een nadrukkelijke structuur dat heeft het laatste</hypothesis>
 <tc1>00:00:00:00</tc1>
 <tco>00:00:00:00</tco>
 <score>6.129283</score>
 </title>
 <title nr="0057">
 <caption>En daar kun je de details mee schilderen.</caption>
 <hypothesis>beetje verf op je palet want er is</hypothesis>
 <tc1>00:00:00:00</tc1>
 <tco>00:00:00:00</tco>
 <score>1.6674248</score>
 </title>
 <title nr="0058">
 <caption>Er zijn ook bredere kwasten. Dat zijn spalters.</caption>
 <hypothesis>is zo glad is niet dat werk bijna</hypothesis>
 <tc1>00:00:00:00</tc1>
 <tco>00:00:00:00</tco>
 <score>5.586089</score>
 </title>
 <title nr="0059">
 <caption>Dat is geschikt om een groot vlak in te wassen.</caption>
 <hypothesis>en heeft niet echt een nadrukkelijke structuur dat heeft het</hypothesis>

<tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>4.8777437</score>
 </title>
 <title nr="0060">
 <caption>Ook erg fijn. Hier heb ik er nog eentje.</caption>
 <hypothesis>hier deze moet werkt eerst ga er op het</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>6.129283</score>
 </title>
 <title nr="0061">
 <caption>Een enorme kwast met een groot reservoir hier.</caption>
 <hypothesis>laat ik even verschillende soorten papier zien hier</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>3.5739913</score>
 </title>
 <title nr="0062">
 <caption>Waar je veel verf in kunt doen en flinke vakken kunt schilderen.</caption>
 <hypothesis>de verf veel beter verdeelt zich mooi en heeft niet echt een</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>7.1478834</score>
 </title>
 <title nr="0063">
 <caption>En een prachtig dun puntje voor fijne details.</caption>
 <hypothesis>nog een veel water laat ik even verschillende</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>5.5339785</score>
 </title>
 <title nr="0064">
 <caption>Eigenlijk kun je met EEN goede kwast je hele aquarel schilderen.</caption>
 <hypothesis>papier wel die ziet aan mijn kwast ziet ander de structuur</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>5.5339785</score>
 </title>
 <title nr="0065">
 <caption>Ik heb er wat papier bijgepakt.</caption>
 <hypothesis>palet want er is maar heel</hypothesis>
 <tci>00:00:00:00</tci>
 <tco>00:00:00:00</tco>
 <score>6.38529</score>
 </title>
 <title nr="0066">
 <caption>Dat is ook belangrijk voor het maken van een aquarel.</caption>
 <hypothesis>beter verdeelt zich mooi en heeft niet echt een nadrukkelijke</hypothesis>
 <tci>00:04:33:46</tci>
 <tco>00:04:37:60</tco>
 <score>27.120642</score>
 </title>
 <title nr="0067">
 <caption>Ik heb drie verschillende soorten papier liggen.</caption>
 <hypothesis>ga er op het van papier dat</hypothesis>
 <tci>00:04:35:54</tci>
 <tco>00:04:38:44</tco>
 <score>16.176481</score>
 </title>
 <title nr="0068">
 <caption>Er is heel veel aquarelpapier in de handel.</caption>
 <hypothesis>absorbeert de verf veel beter verdeelt zich mooi</hypothesis>
 <tci>00:04:38:63</tci>
 <tco>00:04:41:94</tco>
 <score>23.327452</score>
 </title>
 <title nr="0069">
 <caption>Duur papier, goedkoop papier, dik papier, dun papier.</caption>
 <hypothesis>een nadrukkelijke structuur dat heeft het laatste papier</hypothesis>
 <tci>00:04:44:59</tci>
 <tco>00:04:47:90</tco>
 <score>35.305447</score>
 </title>
 <title nr="0070">
 <caption>Glad papier, grof papier.</caption>
 <hypothesis>soorten papier zien hier</hypothesis>
 <tci>00:04:48:12</tci>
 <tco>00:04:49:78</tco>
 <score>17.816618</score>
 </title>
 <title nr="0071">
 <caption>Koop gewoon wat, laat je adviseren en begin.</caption>
 <hypothesis>verf veel beter verdeelt zich mooi en heeft</hypothesis>

```

<tc>00:04:51:08</tc>
<tc>00:04:54:39</tc>
<score>28.001202</score>
</title>
<title nr="0072">
<caption>Ik heb drie verschillende soorten liggen.</caption>
<hypothesis>ik even verschillende soorten papier zien</hypothesis>
<tc>00:04:54:48</tc>
<tc>00:04:56:96</tc>
<score>16.176481</score>
</title>
<title nr="0073">
<caption>Om te laten zien wat er mogelijk is.</caption>
<hypothesis>dat is niet werkt is zo glad is</hypothesis>
<tc>00:04:57:25</tc>
<tc>00:05:00:56</tc>
<score>15.68548</score>
</title>
<title nr="0074">
<caption>Je neemt een klein beetje verf op je palet.</caption>
<hypothesis>extra zich een geen niet mee begin met een</hypothesis>
<tc>00:04:59:27</tc>
<tc>00:05:02:99</tc>
<score>26.19086</score>
</title>
<title nr="0075">
<caption>Want er is maar heel weinig aquarelverf nodig.</caption>
<hypothesis>wolkenlucht op zo maar extra zich een geen</hypothesis>
<tc>00:05:05:08</tc>
<tc>00:05:08:39</tc>
<score>30.71856</score>
</title>
<title nr="0076">
<caption>En veel water.</caption>
<altcaption>en veel</altcaption>
<hypothesis start="307.33">nog een</hypothesis>
<tc>00:05:07:33</tc>
<tc>00:05:08:57</tc>
<score>4.066089</score>
</title>
<title nr="0077">
<caption>Dan laat ik even verschillende soorten papier zien.</caption>
<hypothesis>niet mee begin met een mat papier dat</hypothesis>
<tc>00:05:08:16</tc>
<tc>00:05:11:47</tc>
<score>33.391205</score>
</title>
<title nr="0078">
<caption>Dan kun je zien hoe het werkt.</caption>
<altcaption>je zien hoe het werkt</altcaption>
<hypothesis start="313.47">zien hier deze moet werkt</hypothesis>
<tc>00:05:13:47</tc>
<tc>00:05:16:37</tc>
<score>4.5025</score>
</title>
<title nr="0079">
<caption>Eerst verf ik op het gladde papier.</caption>
<hypothesis>niet mee begin met een mat papier</hypothesis>
<tc>00:05:16:99</tc>
<tc>00:05:19:89</tc>
<score>12.487545</score>
</title>
<title nr="0080">
<caption>Dat is heet geperst.</caption>
<altcaption>INS INS INS dat is heet geperst</altcaption>
<hypothesis start="318.05">het van papier dat is niet werkt</hypothesis>
<tc>00:05:18:05</tc>
<tc>00:05:19:71</tc>
<score>4.8777437</score>
</title>
<title nr="0081">
<caption>Het is zo glad dat je verf bijna uitglijdt.</caption>
<hypothesis>wolkenlucht op zo maar extra zich een geen niet</hypothesis>
<tc>00:05:21:83</tc>
<tc>00:05:25:55</tc>
<score>14.279103</score>
</title>
<title nr="0082">
<altcaption>No alternative caption found!</altcaption>
<caption>Het kan mooi zijn.</caption>
<hypothesis>kan een hele zijn</hypothesis>
<tc>00:05:25:55</tc>
<tc>00:05:30:64</tc>
<score>8.1</score>

```

```

</title>
<title nr="0083">
  <caption>Maar veel mensen hebben liever dit papier.</caption>
  <hypothesis>niet mee begin met een mat papier</hypothesis>
  <tci>00:05:30:64</tci>
  <tco>00:05:33:54</tco>
  <score>29.074043</score>
</title>
<title nr="0084">
  <caption>Het matte papier. Dat absorbeert de verf veel beter.</caption>
  <hypothesis>met dat water dat drie met je maakt nat</hypothesis>
  <tci>00:05:32:88</tci>
  <tco>00:05:36:60</tco>
  <score>33.810017</score>
</title>
<title nr="0085">
  <caption>En verdeelt zich mooi.</caption>
  <hypothesis>en als je dan</hypothesis>
  <tci>00:05:36:19</tci>
  <tco>00:05:37:85</tco>
  <score>19.387627</score>
</title>
<title nr="0086">
  <caption>En heeft niet echt een nadrukkelijke structuur.</caption>
  <hypothesis>je maakt nat met een sponsje je</hypothesis>
  <tci>00:05:40:61</tci>
  <tco>00:05:43:51</tco>
  <score>31.919876</score>
</title>
<title nr="0087">
  <caption>Dat heeft het laatste papier wel.</caption>
  <hypothesis>niet al te gestructureerde papier zo</hypothesis>
  <tci>00:05:43:50</tci>
  <tco>00:05:45:98</tco>
  <score>24.138908</score>
</title>
<title nr="0088">
  <altcaption>No alternative caption found!</altcaption>
  <caption>Je ziet het al aan mijn kwast.</caption>
  <hypothesis>je hem zo rondom vast op een</hypothesis>
  <tci>00:05:45:98</tci>
  <tco>00:05:48:25</tco>
  <score>8.1</score>
</title>
<title nr="0089">
  <caption>Je ziet de structuur van het papier er doorheen komen.</caption>
  <hypothesis>is ziet dat er zo uit die zien mooi glad</hypothesis>
  <tci>00:05:48:25</tci>
  <tco>00:05:52:39</tco>
  <score>19.813082</score>
</title>
<title nr="0090">
  <caption>Het kan leuk zijn voor een landschap of wolkenlucht.</caption>
  <hypothesis>die zien mooi glad voor is lekkere hele met</hypothesis>
  <tci>00:05:52:82</tci>
  <tco>00:05:56:54</tco>
  <score>22.694468</score>
</title>
<title nr="0091">
  <caption>Maar begin er niet mee. Begin met een mat papier.</caption>
  <hypothesis>heb het kant en klaar blok met aquarelvelletjes en zitten</hypothesis>
  <tci>00:05:56:66</tci>
  <tco>00:06:00:80</tco>
  <score>27.131474</score>
</title>
<title nr="0092">
  <caption>Dat is makkelijk en werkt prettig.</caption>
  <hypothesis>voor is lekkere hele met water</hypothesis>
  <tci>00:05:59:15</tci>
  <tco>00:06:01:63</tco>
  <score>10.036478</score>
</title>
<title nr="0093">
  <caption>En je kunt altijd nog een ander papier uitproberen.</caption>
  <hypothesis>en zitten rondom vast heeft zodat het ook mooi</hypothesis>
  <tci>00:06:03:79</tci>
  <tco>00:06:07:51</tco>
  <score>17.063374</score>
</title>
<title nr="0094">
  <caption>Maar begin met het matte, niet al te gestructureerde papier.</caption>
  <hypothesis>papier is ook heel makkelijk om mee te nemen wat</hypothesis>
  <tci>00:06:07:17</tci>
  <tco>00:06:11:31</tco>

```

<score>32.56975</score>
</title>
<title nr="0095">
<caption>Een los velletje moet even opgespannen worden.</caption>
<hypothesis>een aquarelblok zo als hier heb het</hypothesis>
<tci>00:06:11:88</tci>
<tco>00:06:14:78</tco>
<score>38.796238</score>
</title>
<title nr="0096">
<caption>Als het te dun is gaat het snel bobbelen met dat water.</caption>
<hypothesis>om op te werken samen altijd doen en je het overheen hele</hypothesis>
<tci>00:06:14:34</tci>
<tco>00:06:19:31</tco>
<score>22.57893</score>
</title>
<title nr="0097">
<caption>Je maakt hem nat met een sponsje.</caption>
<hypothesis>en als je klaar met uit dat</hypothesis>
<tci>00:06:19:41</tci>
<tco>00:06:22:31</tco>
<score>16.000694</score>
</title>
<title nr="0098">
<caption>En je pakt dit papieren plakband erbij.</caption>
<hypothesis>als je klaar met uit dat heeft</hypothesis>
<tci>00:06:20:82</tci>
<tco>00:06:23:72</tco>
<score>32.885807</score>
</title>
<title nr="0099">
<caption>Dan tape je hem zo rondom vast op een plankje.</caption>
<hypothesis>doen en je het overheen hele kleine voor ie broekzak</hypothesis>
<tci>00:06:24:39</tci>
<tco>00:06:28:53</tco>
<score>33.681065</score>
</title>
<title nr="0100">
<caption>En als hij opgedroogd is ziet dat er zo uit.</caption>
<hypothesis>en daar kun je zelfs een ansichtkaartje van versturen klein</hypothesis>
<tci>00:06:28:57</tci>
<tco>00:06:32:71</tco>
<score>40.79232</score>
</title>
<title nr="0101">
<caption>Dan is-ie mooi glad geworden en kun je er eindeloos met water opgaan.</caption>
<hypothesis>niks kijk eens wat ontzettend veel verschillende kleuren je daarmee kan maar en gaat</hypothesis>
<tci>00:06:30:72</tci>
<tco>00:06:36:51</tco>
<score>10.217049</score>
</title>
<title nr="0102">
<caption>En als-ie droog wordt spant-ie zichweer helemaal strak op deze manier.</caption>
<hypothesis>kijk eens hoe je kwast werkt nemen eenvoudig voorwerpje ik heb deze twee</hypothesis>
<tci>00:06:37:12</tci>
<tco>00:06:42:50</tco>
<score>69.82095</score>
</title>
<title nr="0103">
<caption>Het is wat bewerkelijk en veel mensen...</caption>
<hypothesis>werken samen altijd doen en je het</hypothesis>
<tci>00:06:41:05</tci>
<tco>00:06:43:95</tco>
<score>18.407043</score>
</title>
<title nr="0104">
<caption>kopen daarom liever een aquarelblok.</caption>
<hypothesis>als heb je een mooi</hypothesis>
<tci>00:06:44:57</tci>
<tco>00:06:46:64</tco>
<score>14.1485405</score>
</title>
<title nr="0105">
<altcaption>No alternative caption found!</altcaption>
<caption>Zoals ik hier heb.</caption>
<hypothesis>met mensen als heb</hypothesis>
<tci>00:06:46:64</tci>
<tco>00:06:48:43</tco>
<score>8.1</score>
</title>
<title nr="0106">
<caption>Dat is een kant en klaar blok met aquarelvelletjes.</caption>
<hypothesis>staand neem ultramarijn cadmiumrood en cadmiumgeel meng eens wat</hypothesis>
<tci>00:06:48:43</tci>

<tco>00:06:52:15</tco>
 <score>30.057247</score>
 </title>
 <title nr="0107">
 <caption>Die zitten rondom vast gelijmd.</caption>
 <hypothesis>wat zitten zijn laat ook</hypothesis>
 <tci>00:06:52:69</tci>
 <tco>00:06:54:76</tco>
 <score>18.387848</score>
 </title>
 <title nr="0108">
 <caption>Zodat het ook strak blijft als het nat wordt met water.</caption>
 <hypothesis>aquarelletjes net geschilderd van een eenvoudig om het je met rood</hypothesis>
 <tci>00:06:55:96</tci>
 <tco>00:07:00:51</tco>
 <score>33.096745</score>
 </title>
 <title nr="0109">
 <caption>En als je klaar bent snij je het los.</caption>
 <hypothesis>om het je met rood geel en daar anders</hypothesis>
 <tci>00:06:59:14</tci>
 <tco>00:07:02:86</tco>
 <score>11.474178</score>
 </title>
 <title nr="0110">
 <caption>Dan heb je een mooi glad papier.</caption>
 <hypothesis>het maken van een aquarel mijn advies</hypothesis>
 <tci>00:07:01:71</tci>
 <tco>00:07:04:61</tco>
 <score>17.79819</score>
 </title>
 <title nr="0111">
 <caption>Dit is ook makkelijk mee te nemen.</caption>
 <hypothesis>kijk eens hoe je kwast werkt nemen</hypothesis>
 <tci>00:07:04:78</tci>
 <tco>00:07:07:68</tco>
 <score>13.836149</score>
 </title>
 <title nr="0112">
 <altcaption>No alternative caption found!</altcaption>
 <caption>Er zit namelijk een beschermlaagje overheen.</caption>
 <hypothesis>het maken van een aquarel mijn</hypothesis>
 <tci>00:07:07:68</tci>
 <tco>00:07:09:69</tco>
 <score>8.1</score>
 </title>
 <title nr="0113">
 <caption>Ze zijn er in allerlei soorten en maten.</caption>
 <hypothesis>om het je met rood geel en daar</hypothesis>
 <tci>00:07:09:69</tci>
 <tco>00:07:13:00</tco>
 <score>27.574251</score>
 </title>
 <title nr="0114">
 <caption>Hier heb ik nog een lekkere grote.</caption>
 <hypothesis>aquarelletjes net geschilderd van een eenvoudig om</hypothesis>
 <tci>00:07:13:46</tci>
 <tco>00:07:16:36</tco>
 <score>26.409628</score>
 </title>
 <title nr="0115">
 <caption>Heerlijk om op te werken.</caption>
 <hypothesis>eens wat op papieren kijk</hypothesis>
 <tci>00:07:15:56</tci>
 <tco>00:07:17:63</tco>
 <score>23.570284</score>
 </title>
 <title nr="0116">
 <caption>En hier is ook nog een leuke kleine voor in je broekzak.</caption>
 <hypothesis>echte werk in de opzetten van het in in we je een</hypothesis>
 <tci>00:07:17:29</tci>
 <tco>00:07:22:26</tco>
 <score>9.851749</score>
 </title>
 <title nr="0117">
 <caption>Daar kun je zelfs een ansichtkaartje van versturen.</caption>
 <hypothesis>zijn laat je adviseren denk niet dat de</hypothesis>
 <tci>00:07:24:00</tci>
 <tco>00:07:27:31</tco>
 <score>34.09462</score>
 </title>
 <title nr="0118">
 <altcaption>No alternative caption found!</altcaption>
 <caption>Zo'n klein aquarelletje.</caption>

<hypothesis></hypothesis>
<tci>00:07:27:31</tci>
<tco>00:07:32:26</tco>
<score>8.1</score>
</title>
<title nr="0119">
<caption>We hebben gezien wat we nodig hebben voor een aquarel.</caption>
<hypothesis>het water werkt wat de kwasten doen volgende keer gaan</hypothesis>
<tci>00:07:32:26</tci>
<tco>00:07:36:40</tco>
<score>8.4985695</score>
</title>
<title nr="0120">
<caption>Mijn advies is om eerst drie kleuren te kopen.</caption>
<hypothesis>namelijk dat is niet zo maar koop wel een</hypothesis>
<tci>00:07:36:84</tci>
<tco>00:07:40:56</tco>
<score>19.18843</score>
</title>
<title nr="0121">
<caption>Neem ultramarijn, cadmiumrood en cadmiumgeel.</caption>
<hypothesis>daarmee kan maar en gaat</hypothesis>
<tci>00:07:41:59</tci>
<tco>00:07:43:66</tco>
<score>28.620056</score>
</title>
<title nr="0122">
<caption>Meng eens wat op papieren. Kijk eens hoe je kwast werkt.</caption>
<hypothesis>in de opzetten van het in in we je een ander</hypothesis>
<tci>00:07:46:76</tci>
<tco>00:07:51:31</tco>
<score>51.01949</score>
</title>
<title nr="0123">
<caption>Neem een eenvoudig voorwerpje.</caption>
<hypothesis>doen een aan mijn</hypothesis>
<tci>00:07:49:30</tci>
<tco>00:07:50:96</tco>
<score>14.279103</score>
</title>
<title nr="0124">
<caption>Ik heb deze twee aquarelletjes net geschilderd.</caption>
<hypothesis>lekkere experimenteer deze week probeer verf uit</hypothesis>
<tci>00:07:52:42</tci>
<tco>00:07:55:32</tco>
<score>39.608753</score>
</title>
<title nr="0125">
<caption>Met rood, geel en blauw. Anders niks.</caption>
<hypothesis>met het echte werk in de opzetten</hypothesis>
<tci>00:07:56:37</tci>
<tco>00:07:59:27</tco>
<score>30.652552</score>
</title>
<title nr="0126">
<caption>Kijk eens hoeveel kleuren je daarmee kunt maken.</caption>
<hypothesis>het in in we je een ander ziet</hypothesis>
<tci>00:07:59:83</tci>
<tco>00:08:03:14</tco>
<score>13.083445</score>
</title>
<title nr="0127">
<caption>Probeer dat eens uit van de week.</caption>
<hypothesis>ziet dat om maar duidelijk behoefte ander</hypothesis>
<tci>00:08:02:71</tci>
<tco>00:08:05:61</tco>
<score>10.116831</score>
</title>
<title nr="0128">
<caption>Ontzettend leuk om te doen.</caption>
<hypothesis>werkt wat de kwasten doen</hypothesis>
<tci>00:08:04:51</tci>
<tco>00:08:06:58</tco>
<score>25.70012</score>
</title>
<title nr="0129">
<caption>Mijn tip voor deze keer is:</caption>
<hypothesis>de kwasten doen volgende keer gaan</hypothesis>
<tci>00:08:35:26</tci>
<tco>00:08:37:74</tco>
<score>28.776392</score>
</title>
<title nr="0130">
<caption>Ga naar een speciaalzaak, laat je adviseren.</caption>


```

<hypothesis>we je een ander ziet dat om</hypothesis>
<tci>00:08:37:42</tci>
<tco>00:08:40:32</tco>
<score>12.498631</score>
</title>
<title nr="0131">
<caption>Denk niet dat je heel veel mooie spullen nodig hebt.</caption>
<hypothesis>in in we je een ander ziet dat om maar</hypothesis>
<tci>00:08:40:73</tci>
<tco>00:08:44:87</tco>
<score>14.619331</score>
</title>
<title nr="0132">
<caption>Dat is niet zo,maar koop wel een paar goede dingen.</caption>
<hypothesis>opzetten van het in in we je een ander ziet dat</hypothesis>
<tci>00:08:44:89</tci>
<tco>00:08:49:44</tco>
<score>46.210495</score>
</title>
<title nr="0133">
<caption>Experimenteer lekker.</caption>
<hypothesis>experimenteer deze</hypothesis>
<tci>00:00:00:00</tci>
<tco>00:00:00:00</tco>
<score>7.1395516</score>
</title>
<title nr="0134">
<caption>Probeer de verf uit.</caption>
<hypothesis>probeer wat de met</hypothesis>
<tci>00:00:00:00</tci>
<tco>00:00:00:00</tco>
<score>8.076491</score>
</title>
<title nr="0135">
<caption>Probeer wat de kleuren met elkaar doen.</caption>
<hypothesis>doen volgende keer gaan met het echte</hypothesis>
<tci>00:08:50:95</tci>
<tco>00:08:53:85</tco>
<score>14.695535</score>
</title>
<title nr="0136">
<caption>Hoe het water werkt, wat de kwasten doen.</caption>
<hypothesis>hoe het water werkt wat de kwasten doen</hypothesis>
<tci>00:08:53:56</tci>
<tco>00:08:56:87</tco>
<score>26.999107</score>
</title>
<title nr="0137">
<caption>Volgende keer beginnen we met het echte werk:</caption>
<hypothesis>het in in we je een ander ziet</hypothesis>
<tci>00:08:55:47</tci>
<tco>00:08:58:78</tco>
<score>15.716294</score>
</title>
<title nr="0138">
<caption>Het opzetten van een aquarel van een stillevens.</caption>
<hypothesis>in we je een ander ziet dat om</hypothesis>
<tci>00:08:59:09</tci>
<tco>00:09:02:40</tco>
<score>10.116831</score>
</title>
<title nr="0139">
<altcaption>No alternative caption found!</altcaption>
<caption>De actuele onderwerpen zijn zinnige onderwerpen.</caption>
<hypothesis></hypothesis>
<tci>00:09:02:40</tci>
<tco>00:09:09:83</tco>
<score>8.1</score>
</title>
<title nr="0140">
<caption>Waar duidelijk behoefte aan is.</caption>
<hypothesis>maar duidelijk behoefte ander hele</hypothesis>
<tci>00:09:09:83</tci>
<tco>00:09:11:90</tco>
<score>14.279103</score>
</title>
<title nr="0141">
<caption>MAX moet blijven bestaan.</caption>
<hypothesis></hypothesis>
<tci>00:00:00:00</tci>
<tco>00:00:00:00</tco>
<score>0</score>
</title>
<title nr="0142">

```

<caption>Ze moeten een stem hebben in Hilversum.</caption>
<hypothesis></hypothesis>
<tc1>00:00:00:00</tc1>
<tco>00:00:00:00</tco>
<score>0</score>
</title>
</titles>

AI	Artificial Intelligence, 11
AM	Acoustic Model, 27
API	Application Programmable Interface, 77
ASR	Automatic Speech Recognition, 15
CC	Closed Captions, 33
CGN	Corpus Gesproken Nederlands, 88
DOS	MicroSoft Disk Operating System, 80
DOSEMU	DOS Emulation, 80
ERD	Entity Relationship Diagram, 57
FPA	First-Pass Alignment, 70
FSM	Finite State Machine, 25
FTP	File Transfer Protocol, 91
GHC	Glasgow Haskell Compiler, 80
GUI	Graphical User Interface, 91
HMM	Hidden Markov Model, 25
HOS	Humanistische Omroep, 43
IR	Information Retrieval, 83
KBS	Knowledge-Based System, 31
KRO	Katholieke Radio Omroep, 48
LD	Levenshtein Distance, 71
LPC	Linear Predictive Coding, 23
LVCSR	Large-Vocabulary Continuous Speech Recognition, 16
ML	Maximum Likelihood, 27
MT	Machine Translation, 33
NPO	Nederlandse Publieke Omroep, 15
NPS	Nederlandse Programma Stichting, 45
OCR	Optical Character Recognition, 63

PAL	Phase Alternating Line, 13
PCM	Pulse Code Modulation, 55
PCM	Pulse-Code Modulation, 79
Perl	Practical Extraction Report Language, 91
PHP	PHP Hypertext Preprocessor, 91
SCP	Secure Copy, 91
SECAM	Séquentiel Couleur à Mémoire, 13
SL	Source Language, 33
SLM	Statistical Language Model, 26
SOX	SOund eXchange, 79
SPA	Second-Pass Alignment, 70
SSH	Secure shell, 91
STL	Spruce Subtitle Format, 91
TCI	Time Code In, 90
TCO	Time Code Out, 90
TF-IDF	Term Frequency Inverse Document Frequency, 66
TL	Target Language, 33
TNO	Nederlandse organisatie voor Toegepast-Natuurwetenschappelijk Onderzoek, 80
TTS	Text-to-Speech, 24
URI	Uniform Resource Locator, 91
VM	Virtual Machine, 91
VPRO	Vrijzinnig Protestantse Radio Omroep, 39
XML	eXtensible Markup Language, 86

- [Ahmer02] I. Ahmer
Automatic Speech Recognition for Closed Captioning of Television:
Data and Issues
Univerity of South Australia, 2002
- [Alphen92] P. van Alphen
HMM-based continuous-speech recognition, pp. 55-111
Universiteit van Amsterdam, 1992
- [Antonsen08] L. Antonsen
Sentence Alignment of New Testament in two sámi languages
University of Tromsø, 2008
- [Boekee91] D.E. Boekee and J.C.A. van der Lubbe
Informatietheorie, tweede druk
Delftse Uitgevers Maatschappij, 1991
- [Boogaard07] J. Boogaard
Dialogue Strategies, Spoken Language Understanding and Call Clas-
sification
Literature Study at Delft University of Technology, 2007
- [Chen99] S. F. Chen & J. Goodman
An Empirical Study of Smoothing Techniques for Language Modeling
Harvard University, 1999
- [Chen04] Chen, Lamel and Gauvain
Lightly Supervised Acoustic Model training Using Consensus Net-
works
Spoken Language Processing Group at LIMSI, France, 2004
- [Frenz05] M. Frenz
Pro Perl Parsinge
Apress, 2005
- [Gamma95] E. Gamma, R. Helm, R. Johnson and J. Vlissides
Design Patterns - Elements of Reusable Object-Oriented Software
Addison-Wesley, 1995
- [Ghezzi91] C. Ghezzi, M. Jazayeri, D. Mandrioli
Fundamentals of Software Engineering
Prentice Hall, 1991

- [Gibbon97] D. Gibbon, R. Moore, R. Winski
Handbook of Standards and Resources for Spoken Language Systems, pp.237-244
Walter de Gruyter & Co, 1997
- [Goldsmith93] S. Goldsmith
A practical guide to real-time systems development
Prentice Hall Europe, 1993
- [Gornik] D. Gornik
A technical discussion on modeling with UML
IBM, 2003
- [Heuvel63] H. van den Heuvel
Speaker variability in acoustic properties
Katholieke Universiteit Nijmegen, 1963
- [Huang03] C. Huang
Automatic Closed Caption Alignment
Colombia, 2003
- [Imai07] T. Imai, S. Homma, A. Kobayashi, S. Sato, T. Takagi, K. Saitou and S. Hara
Real-Time Closed-Captioning Using Speech Recognition
Japan Broadcasting Corporation, 2007
- [Jongebloed08] H. Jongebloed
Nulmeting alignment ondertiteling Nederlandse Publieke Omroep
Dutcheer, 2008
- [Jurafsky00] D. Jurafsky and J. H. Martin Speech and Language Processing, International Edition, pp. 66-67
Prentice-Hall, 2000
- [Khan06] N. Khan, T. Habib, J. Alam, R. Rahman, N. Uzzaman, M. Khan
History (forward N-gram) or Future (backward N-gram)?
Which model to consider for N-gram analysis in Bangla?
BRAC University, Bangladesh, 2006
- [Klerk06] A. de Klerk
Keyword Identification for Service-Desk Call Classification
2006
- [Langhorst03] A. Langhorst and M. Steinle
Pipes and Filters Architectural Pattern
Hasso-Plattner-Institute for Software Systems Engineering, 2003

- [Lee93] C.H. Lee, J.L. Gauvain, R. Pieraccini and L. R. Rabiner
Large vocabulary speech recognition using subword Units
AT&T Bell Laboratories, New Jersey, 1993
- [Li00] J. Li
Hidden Markov Model
The Pennsylvania State University, 2000
- [Maganti07] H. K. Maganti, P. Motlicek and D. Gatica-Perez
Unsupervised speech/non-speech detection for automatic speech
recognition in meeting rooms
IDIAP Research Institute, University of Ulm, Ecole Polytechnique
Federale de Lausanne, 2007
- [Mitchell01] M. Mitchell, J. Oldham and A. Samuel
Advanced Linux Programming
New Riders Publishing, First Edition, 2001
- [Mittelbach07] F. Mittelbach and M. Goossens
The L^AT_EX Companion, Second Edition
Addison-Wesley, 2007
- [Negnevitsky02] M. Negnevitsky
Artificial Intelligence, A Guide to Intelligent Systems, pp. 26-28
Addison-Wesley, 2002
- [Norton00] P. Norton
Peter Noton's Complete Guide to Linux
Academic Service, 2000
- [Nye84] H. Nye
The Use of a One-Stage Dynamic Programming Algorithm for Con-
nected Word Recognition
IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984
- [Olson98] E. Olson
Programming with Perl Modules
O'Reilly & Associates, 1998
- [Pellom02] B. Pellom
Sonic: Digit Recognition Tutorial University of Colorado, 2002
- [Pellom04] B. Pellom and K. Hacioglu
SONIC: The University of Colorado Continuous Speech Recognizer
University of Colorado, 2004

- [Rabiner93] L. Rabiner and B. Juang
Fundamentals of Speech Recognition
Prentice Hall, 1993
- [Ramos03] J. Ramos
Using TF-IDF to Determine Word Relevance in Document Queries
University of Groningen, 2008
- [Robert-Ribes98] J. Robert-Ribes
On the use of automatic speech recognition for TV captioning
CSIRO-MIS and ACSys, North Ryde, 1998
- [Seyerlehner07] K. Seyerlehner, T. Pohle, M. Schedl, G. Widmer
Automatic music detection in television productions
Johannes Kepler University Linz, Austria, 2007
- [Son07] R.J.J.H. van Son and D. Weenink
Speech recognition and synthesis
Universiteit van Amsterdam, Phonetic Sciences, 2007
- [Thompson99] S. Thompson
Haskell, The Craft of Functional Programming
Second edition, Addison Wesley, 1999
- [Tiedemann07] J. Tiedmann
Improved Sentence Alignment for Movie Subtitles
University of Groningen, 2008
- [Tiedemann08] J. Tiedmann
Synchronizing Translated Movie Subtitles
University of Groningen, 2008
- [Wang02] J. Wang, T. Chua
A Framework for video Scene Boundary Detection
School of Computing, Singapore, 2002
- [White96] S. A. White
Software Architecture Design Domain
University of Houston, 1996
- [Wiggers08] P. Wiggers
Modelling Context in Automatic Speech Recognition, pp. 17-28
Technische Universiteit Delft, 2008
- [Woodland97] P.C. Woodland, M.J.F. Gales, D. Pye, S.J. Young
Broadcast news transcription using HTK
Cambridge University Engineering Department, England, 1997

- [Young97] S. Young and G. Bloothoof
Corpus-Based Methods in Language and Speech Processing, pp. 29-
34
Kluwer Academic Publishers, 1997
- [Yourdon89] E. Yourdon
Modern Structured Analysis
Prentice-Hall, 1989

Web Resources

- [7zip] <http://www.7-zip.org/>
7-Zip
visited on January 19, 2010
- [Audacity] <http://sourceforge.net/projects/audacity/>
SourceForge.net:Audacity
visited on April 8, 2009
- [CMU-CAM] <http://www.speech.cs.cmu.edu/SLM/toolkit.html>
The CMU-Cambridge Statistical Language Modeling Toolkit v2
visited on August 17, 2009
- [DOSEMU] <http://www.dosemu.org/>
DOSEMU
visited on January 27, 2010
- [Emerson] http://www.emerson.emory.edu/services/latex/latex_toc.html
LaTeX help 1.1
visited on April 5, 2009
- [FFmpeg] <http://www.ffmpeg.org/>
FFmpeg
visited on April 6, 2009
- [Graphviz] <http://www.graphviz.org/>
Graphviz - Graph Visualization Software
visited on January 28, 2010
- [GHC] <http://www.haskell.org/ghc/>
The Glasgow Haskell Compiler
visited on December 10, 2009
- [Haskell] <http://haskell.org/>
Haskell
visited on April 6, 2009
- [Hexedit] <http://www.physics.ohio-state.edu/~prewett/hexedit/>
Hexedit
visited on April 8, 2009
- [Hypertext] <http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/teTeX/latex/>
Hypertext Help with LaTeX
visited on April 5, 2009

- [Kronto] <http://www.kronto.org/thesis/tips/index.html>
General LaTeX tips and tricks
visited on April 1, 2009
- [Spectrogram] <http://www.wilhelm-kurz-software.de/dynaplot/applicationnotes/spectrogram.htm>
Wilhelm Kurz Software
visited on May 16, 2009
- [Spectrum] <http://commons.wikimedia.org>
Wikimedia Commons
visited on May 16, 2009
- [Springer] <http://www.springer.com/computer/lncs?SGWID=0-164-7-72376-0>
LNCS Journals, Academic Books & Online Media
visited on April 16, 2009
- [Tofrodos] <http://www.thefreecountry.com/tofrodos/index.shtml>
Tofrodos
visited on January, 2010
- [Perl03] http://envgen.nox.ac.uk/courses/perl_bioperl/
Introduction to Perl modules
visited on April 13, 2009
- [Perl05] <http://stuff.mit.edu/iap/2006/perl/slides/welcome.html>
Welcome to Perl Programming
visited on April 13, 2009
- [Perl09] <http://www.regular-expressions.info/perl.html>
Perl Text Patterns for Search and Replace
visited on April 9, 2009
- [PHP] <http://nl.php.net/>
PHP:Hypertext Preprocessor, visited on April 3, 2009
- [Softni] <http://www.softni.com/subtitler/fileformat.html>
Softni fileformat, visited on April 3, 2009
- [SoX] <http://sox.sourceforge.net/>
SoX - Sound eXchange, visited on April 3, 2009
- [SONIC] <http://www.bltek.com/virtual-teacher-side-menu/sonic.html>
Sonic: Large Vocabulary Continuous Speech Recognition System, visited on January 2010

- [Subversion] <http://subversion.tigris.org/>
Subversion, visited on January 2010
- [Tomcat] <http://tomcat.apache.org/>
Apache Tomcat, visited on February 1, 2010
- [VMware] <http://www.vmware.com/>
VMware Virtualization Software for Desktops, Servers & Virtual Machines for a Private Cloud, visited on February 3, 2010
- [XAMPP] <http://www.apachefriends.org/en/xampp.html>
XAMPP, visited on April 3, 2009
- [XML] <http://www.w3c.nl/Dutch/xml-nl-10punten.shtml>
XML in 10 punten, visited on April 17, 2009
- [Xuggler] <http://www.xuggle.com/xuggler/>
Xuggle Xuggler, visited on February 1, 2010