Karlsruhe Institute of Technology

Automatic Segmentation and Summarization of Spoken Lectures

Master Thesis of

Narine Kokhlikyan

At the faculty of Computer Science Institute for Anthropomatics(IFA)

Advisors:

Prof. Dr. Alex Waibel
Prof. Joy Ying Zhang
M.Sc. Yuqi Zhang

Duration: 01. June 2012 – 30. November 2012

Abstract

The ever-increasing number of online lectures has created an unprecedented opportunity for distance learning. Most online lectures are presented as unstructured text, audio and/or video files which make it difficult for students to locate relevant lectures and browse through them. In this thesis, we investigated several *automatic lecture segmentation* and *summarization* algorithms. *Automatic lecture segmentation* algorithms impose structure by inserting paragraph separators and section titles to the lecture to make them more readable. The *Segmentation* algorithms include K-Means, HMM, Hierarchical Segmentation. Unlike segmentation, *Automatic lecture summarization* algorithms compress lectures by selecting salient information based on importance of words/phrases and provide concise, coherent and readable summaries. The summarization algorithms were developed based on two supervised machine learning approaches: Support Vector Machine (SVM) and Conditional Random Fields (CRF). These approaches show comparable results with existing summarization methods.

In this thesis we propose a novel Rhetorical Structure Index(RSI) to measure the structural importance of a feature. Experiments show that using RSI has significantly improved the segmentation accuracy as compared to the traditional content-based feature weighting scheme such as TF/IDF. We also build summarization systems with structure-dependent models such that summarization generated for "Introduction" is different from that for "Main Lecture Content" and "Conclusion". From all three segmentation algorithms HMM showed the best performance. However, similar to K-Means, HMM has definite number of segments which is not the case for Hierarchical Segmentation. Experiments show that using structure-dependent summarization outperforms the uniform summarization method by 15%.

I hereby declare that this thesis is my own original work which I created without illegitimate help by others, that I have not used any other sources or resources than the ones indicated and that due acknowledgement is given where reference is made to the work of others

Karlsruhe, 30. November 2012 .....

(Narine Kokhlikyan)

Contents

List of Figures

List of Tables

1. Introduction

Online learning, a new trend of distance learning, provides numerous online lectures to students all over the world. More than 19,000 colleges [Coll] all over the world offer thousands of free online lectures. This makes studying more comfortable and attractive. However, the retrieval and indexing of required information become a challenging task. Many automatically transcribed lectures do not own explicit structural information to facilitate navigation or contain irrelevant and repeated information.

Despite some of these challenges have been investigated for a long time in different research areas, little has been done to index, structure and shorten the actual content of lectures, so that students can browse through large collections of lectures to find the ones that interest them. This is because the demands on automatic lecture summarization are qualitatively different from those used for the type of discourse found in broadcast news, for example.

This research pursues an approach that captures the particular structural elements common to lectures in order to arrive at an adequate summary. The focus of this work is to develop an alternative approach that addresses the problems inherent in the lecture by analyzing existing methods for speech summarization and segmentation, adapting and extending some of these general concepts to the lecture format.

1.1 Motivation

As previously mentioned, distance learning has become widely accepted, a fact made possible by the wide availability of audio lecture recordings, their transcriptions and translations in different languages. However, lecture transcriptions or their translations often miss explicit structural information, contain redundant information, missing sentence boundaries and unrecognized words. This presents difficulties finding required information from growing number of lectures. To illustrate some of these problems let's take a look at a small piece of a lecture transcript in English shown on Figure 1.1. Since the example lecture is extensive we omit irrelevant sections indicated by three periods. By looking at the transcript it is easy to recognize

1. *today 's topic is system combination for translation systems*
2. *why are we doing this ?*
3. *first of all , there is a large number of different ways of how automatic translations can be produced .*
4. *there are systems based on rules systems based on examples on statistics on hierarchies some using syntax and so on*

…

5. *i mean that is partly why I chose this example so that you can see that the best system is not the best system for all sentences .*
6. *yes ?*
7. *so you have a system that does something and its output goes into the next system and this translates then again and ultimately you get a translation.*

….

8. *but it is really hard to interpret why it looks like this now because these are all statistic models .*
9. *i think that were all results that I had .*
10. *just so that you have seen this once .*
11. *yes .*

Figure 1.1: An example lecture transcript before segmentation and summarization.

that overall there is no explicit structure, particularly in the lack of paragraphs. Some sentences are redundant or comprise less information about the topic of the talk. For example, the sentences in lines two, six and ten are redundant and do not contain topic-relevant information. Moreover, some of the sentences are grammatically not sound and contain speech disfluency. For example, the sentence in line five is grammatically not sound. Our goal is to penalize and remove those sentences, in-

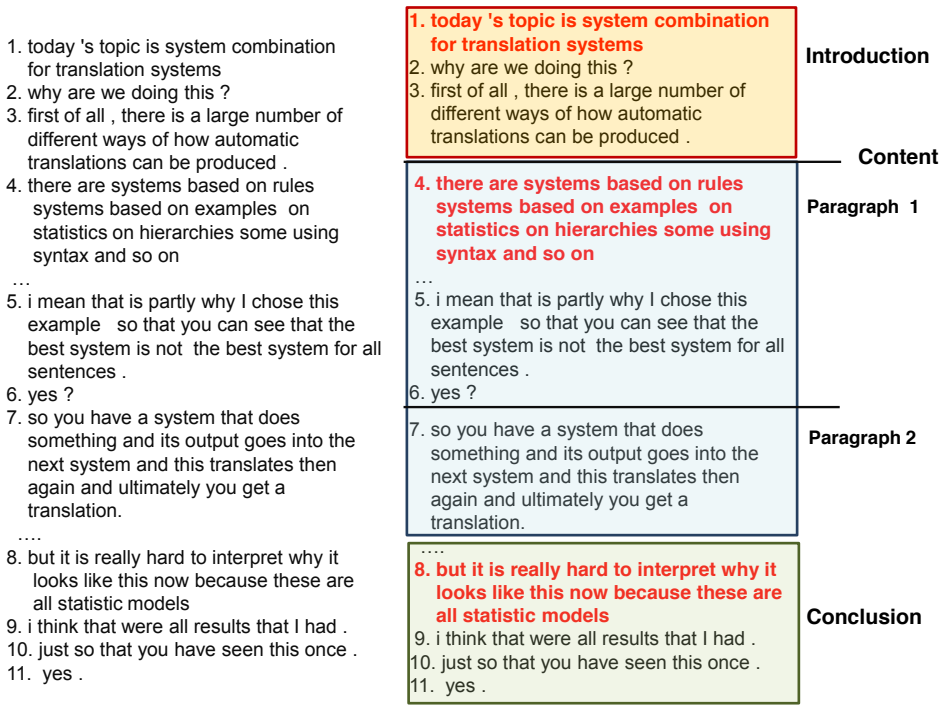

Figure 1.2: An example of automatically recognized lecture before (left side) and after segmentation and summarization (right side). The summary sentences are marked bold red in segments.

sert paragraphs, make the transcripts shorter, readable and concise. On Figure 1.2 on the right side we can see an example summary. It is generated automatically after selecting only 20 percent of all sentences. Sentence selection is based on con-

tent relevant words and phrases. By comparing the content information between the original text and the summary we can recognize that the summary transfers the same message as the original text. The summary is more concise and requires less time for reading and grasping the idea. Moreover, the paragraphs bring in more structure, separate different topics and aid in understanding. The example depicted on Figure 1.2 shows the transcribed lecture before segmentation/summarization on the left side and after on the right side Figure 1.2.

We can observe that the summary of the first paragraph "introduction" selects only one sentence. With this sentence we introduce the topic. The second paragraph, "content", is divided into smaller sub-paragraphs. The summary sentences explain the core idea: the translation systems. The summary of the last paragraph, "conclusion", ignores last three sentences since they are content irrelevant. We can see that the generated summary has paragraphs and comprises only content-relevant information thereby reducing the time needed to read and understand the lectures.

The goal of this thesis is to build a segmentation/summarization system which segments and condenses lectures by removing irrelevant information and considering structural characteristics of the lecture. The system is evaluated and compared with existing solutions such as MMR and OTS, Chori's summarization system which has presented significant improvements.

1.2 Thesis Statement

This thesis aims to develop a system for automatically segmenting and summarizing spoken lectures. The output of our system is concise and coherent summary with proper paragraph boundaries. Extracted summaries contain most relevant and salient information spoken during the lecture. In order to segment lectures the system proposes new metrics and uses well-known statistical models such as Hidden Markov Model(HMM), K-Means and hierarchical segmentation methods. In order to extract salient information from lectures the system uses structural information and Conditional Random Fields(CRF) together with a set of features to learn summarization parameters and generate concise summaries.

In this thesis, specifically, we address to the problems of less structured, redundant information and information retrieval. In order to show that the purposed method can effectively deal with such problems it is compared with different automatic summarization systems such as Maximum Marginal Relevance (MMR) [CaGo98], Open Text Summarizer(OTS) [Rote], Chiori's speech-to-text summarizer [HFMY+02]. Besides, the experiments have revealed that lecture segmentation improves the performance of summarization system by 15% in average.

1.3 Thesis Outline

The organization of this thesis is as follows: Chapter 2 introduces existing technics and tools in automatic text summarization, folowed by related work in summarization and segmentation. Next Chapter 3 describes lecture specific characteristics such as structure, speech discourse, interaction and dialect. Chapter 4 presents entire Segmentation Sumarization System, describes individual components and the

dependencies. Chapter 5 presents implementation tools and details. Chapter 6 illustrates and discusses evaluation results both for segmentation and summarization. Finally, the last Chapter 7 concludes the thesis, presents the contributions and future work.

2. Background and Literature Review

2.1 Summarization

One of the most popular and naive techniques in both text and speech summarization is sentence extraction. Extractive techniques use statistical heuristics and select sentences without changing the content and the meaning. The selection is based on statistical methods and measurement metrics such as TF*IDF and Centroid which rank the sentences based on different criteria. Each sentence is assigned a score and the sentences with the highest score are extracted from the document presenting the summary. However, only sentence extraction presents limitations since it does not allow to change the original text such as add new phrases or delete existing ones thereby causing non-coherence between the extracted sentences. The summarization approaches which consider not only extraction of existing information but also allow paraphrasing the existing content are called abstractive approaches. For example word extraction can be used to create abstractive summaries.

Word extraction, a common technique in speech summarization, allows to remove redundant information or filled pauses by selecting only content words. It penalizes redundant words or filled pauses for example [hmmm, um] which are common in spoken communication. However, sentence extraction still leads word extraction in performance and application areas. Many summariaztion approaches use word extraction as one of the initial steps directed to abstractive summaries. Ideally, abstractive summaries allow not only changing the original sentences but also adding new words and form new phrases. However, abstractive approaches are rare due to complex semantic reasoning problems and ambiguity of natural language.

2.1.1 Text Summarization

The goal of automatic text summarization is to compress text by extracting the most important parts from the text in its entirety. During the last several decades many approaches to automatic text summarization have been developed. The preceding paragraphs introduce some of those approaches.

One of the earliest approaches was developed by Jamie Carbonell and Jade Goldstein (1998)[CaGo98], researchers at Carnegie Mellon University. They introduced Maximum Marginal Relevance (MMR), one of the most popular approaches of information retrieval and text summarization. MMR reduces redundant information by ranking and selecting query-relevant sentences based on similarity measures. Selected sentences have maximum similarity to the query and minimum similarity to previously extracted sentences. This approach and many of its extensions have found wide application not only in text, but also in speech summarization.

Dormir R. Radev et al.(2004) [ErRa11] proposed a very different approach than MMR, based on a similarity graph. They suggest three different methods for calculating the centrality of sentences in a similarity graph in order to identify the most important sentences. The vertices of a similarity graph represent sentences, and the edges represent weighted similarities between those vertices. The first method considers document cluster where centroid of cluster is used to measure similarity. The second method states that the sentences which are similar to many other sentences are more salient. The third method computes a degree of centrality based on the number of similar sentences by using a threshold. However, all of these methods have some disadvantages. The first and third methods depend on the choice of centroid and threshold, the second method considers all similarities to be equally important.

While the approaches described above deal with only one language, there are several approaches dealing with cross-language summarization. These approaches try to use bilingual information to extract the summaries. Cross-language text summarization addresses the problem of summarization in the target language which is derived from the text in the source language. Xiaojun Wan et al. in 2010 [WaLX10] proposed a cross-language document summarization method based on machine translation quality. The proposed summarization method selects sentences based on their informativeness and translation quality. The authors show the effectiveness of their method for English-to-Chinese translation. Later, Xiaojun Wan (2011) [Wan11] suggested two graph-based approaches of cross-language document summarization. One of the approaches, the so-called SimFusion approach, computes the similarity of sentences in the target language based on similarity fusions in source and target languages. Co-Rank, the second approach, considers three types of sentence relationships: similarities between source-to-source, target-to-target and source-to-target sentences. These approaches perform well only when the text contains grammatically correct sentences, sentence boundaries and appropriate organization.

In order to compare the performance of summarization approaches objectively, different evaluation benchmarks have been developed. Both the Document Understanding Conference (DUC) [21] and the Text Analysis Conference(TAC) [22] are annual conferences, providing benchmarks and infrastructure for evaluating summarization systems. The benchmark provided by the DUC offers a large amount of data, mostly from news domains, and common tools and methods to evaluate and compare summarization systems. Many summarization approaches such as those described above use the DUC benchmark for evaluation. The TAC, an outgrowth of the DUC, provides large-scale data-set and evaluation infrastructure not only for summarization systems, but also for Natural Language Processing (NLP) in general.

2.1.1.1 Important Information Extraction Methods

TF*IDF

Term frequency-inverse term frequency(tf*idf) is one of the most popular approaches in information retrieval for measuring the significance of terms in all documents [MaRS08].

Term Frequency measures how often a term occurs in documents. There are different methods for calculating term frequency. The natural method measures the number of times a term occurs in document. Term frequency has following donation: $tf_{t,d}$, where t refers to term and d to document.

$$tf_{t,d} = \frac{f_{t,d}}{F_d} \tag{2.1}$$

where $f_{t,d}$ is the number of term t in document d and F_d is the number of all terms in document d. In contrary to term frequency, inverse term frequency rewards rare terms and is defined in following way:

$$idf_t = \frac{log(N)}{n_i} \tag{2.2}$$

where N is the number of all documents and n_i the number of documents where term i occurs.

Tf*idf score assigned to a term t in document d is the product of tf and idf scores.

$$tf * idf_{t,d} = tf_{t,d} \times idf_t \tag{2.3}$$

Modified Version of TF*IDF

One variant of TF*IDF score is sub-linear scaling of TF score. This variant of TF*IDF score does not assign TF the occurrence of word but the logarithm of occurrence. The TF score is given by the equation:

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d} & if \quad tf_{t,d} > 0 \\ 0 & otherwise \end{cases} \tag{2.4}$$

and in equation 2.3 we replace tf with $wf_{t,d}$

$$wf_{t,d} * idf_{t,d} = wf_{t,d} \times idf_t \tag{2.5}$$

Another variant of TF score is maximum TF normalization which normalizes TF score with each term occurring in document d. For each document d, let $tf_{max}(d) = max_{\tau \in d}$, where τ ranges over all terms in t [MaRS08]. The normalized frequency of each term t in document d is calculated by

$$ntf_{t,d} = a + (1 - a)\frac{tf_{t,d}}{tf_{max}(d)} \tag{2.6}$$

where a is a smoothing term and generally has a value between 0 and 1.
Table 2.1 shows different variants of tf and idf scores. TF normalization approach is an interesting technique, however it is unstable and suffers from hard tuning and the change of term words. It might have outlier terms with unusual large occurrence not representative of the document content.

Table 2.1: Different variants of tf and idf scores.

Term Frequency	Inverse Document Frequency
n (natural) $tf_{t,d}$	n(no) 1
l (logarithm) $1 + log(tf_{t,d})$	$t(idf)\frac{N}{df_t}$
a (augmented) $0.5 + \frac{0.5 \times tf_{t,d}}{max_t(tf_{t,d})}$	p(prob idf) $max\{0, \log \frac{N-df_t}{df_t}\}$
b (boolean) $\begin{cases} 1 & if, tf_{t,d} > 0 \\ 0 & otherwise \end{cases}$	
L (log ave) $\frac{1+\log(tf_{t,d})}{1+\log(ave_{t \in d}(tf_{t,d}))}$	

Centroid based Information Retrieval

Centroid is a bag of words having $TF * IDF$ score higher than given threshold for the given set of documents. The sentences which are closer to centroid, contain more words from the centroid, are considered to be more important. This can be used for extracting important sentences from documents. Centroid vector C can be calculated by the following equation:

$$C = \{w | w_{tf*idf} > \theta\} \tag{2.7}$$

Cosine Similarities

Cosine Similarity [MaRS08] is a metric, frequently used to find the similarities between two documents where each document is represented as a vector of words. Cosine similarity metric calculates the normalized dot product of two documents. The goal is to find the cosine angle between two documents. If cosine similarity is 0 the documents do not share any attributes, that means the angle is 90. If the cosine similarity is 1, the documents are equal, that means the angle is 0. Mathematical formulation of cosine similarity is the following:

$$CosineSim(x, y) = cos(\theta) = \frac{x.y}{||x|| * ||y||} \tag{2.8}$$

2.1.1.2 Summarization Software

In this chapter we introduce existing summarization software which we have evaluated and compared to each other. These software includes Open Text Summarizer(OTS) and Lemur Summarization Toolkit. Both use known text summarization techniques to summarize documents and are language independent.

Open Text Summarizer

Open Text Summarizer(OTS) [Rote] is a simple summarization program which supports many languages and platforms. This summarization program supports around 37 languages and implements a simple algorithm of sentence extraction without using any training data or model. The sentences are extracted based on occurrence of non-functional words. For example let's assume, after removing non-functional words we receive the following list of word occurrences "system"-12, "component"-9, "connection"-6, "history"-1. Sentences containing words "system", "component", "connection" will be considered important and the ones containing only "history"

less important. Furthermore, OTS supports Mac, Windows, Ubuntu, Fedora and many other platforms.

Lemur Summarization Toolkit

Lemur is a project for data analysis, mining information extraction and summarization developed at the Carnegie Mellon University(CMU) [oMUn]. It proposes two general algorithms for automatic summarization. The first algorithms, so called basic summarization, extracts sentences with the highest score. Sentence score is assigned based on frequency of words. The second algorithm, a more complex approach, implements maximum marginal relevance [CaGo98] algorithm for retrieving important sentences. It is originally query based, but can be also used without specifying the query. If the query misses, the system uses the first sentence of an original text as a query. However, the first sentence rarely contains content-relevant information and impacts the quality of summarization negatively.

2.1.2 Speech Summarization

Speech summarization, a relatively young research field, aims to compress spoken text by extracting salient information. The extraction process includes removing unimportant information, speech disfluencies, redundant information, missing sentence boundaries, and non-grammatical sentences all of which make the task of speech summarization a great deal more challenging than text-to-text summarziation.

One of the earliest approaches of spoken language summarization was suggested by Zechner and Waibel[ZeWa00] in 2001. In their work, Zechner and Waibel address automatic summarization of spoken dialogues in unrestricted domains. They use the Maximum Marginal Relevance (MMR) framework to rank and select relevant information based on similarity measure, thereby reducing redundant information. However, Zechner's method is strongly dependant on MMR method and applicable only for dialogues and meetings and it is inappropriate for lectures or news.

While the approach introduced by Zechner and Waibel[ZeWa00] focuses on unrestricted dialog domain, Chiori[HFMY$^+$02] proposed a new method for news domains which uses summarization score. The summarization score constitutes linear combination of four different scores. Eq. 2.9 shows the mathematical representation of summarization score:

$$S(V) = \sum_{m=1}^{M} L(v_m|v_{m-1}, w_{m-2}) + \lambda_I I(v_m) + \lambda_C C(v_m) + \lambda_T T(v_{m-1}, v_m) \qquad (2.9)$$

where M is the number of sentences, L represents 3-gram linguistics score, I significance score, C confidence score and T word concatenation score. Based on summarization score, Chiori's method effectively extracts a set of words which maximize the summarization score by removing redundant and unimportant information. Chiori et.al. [HFMY$^+$02] show the effectiveness of the approach on Japanese broadcast news for different summarization ratios. The disadvantage of this method is the lengthy, time-consuming annotation of speech data which must be extracted by selecting single words and recombining them into a summary. Moreover, the performance strongly depends on training data used in the method.

Meeting summarization is another growing research direction next to news and dialogs. Fei Liu [LiLi09], [LiLi10] in 2009 and 2010 proposed an abstractive meeting summarization approach based on sentence compression and merging. Liu's experiments show that despite sentence compression is promising and provides reasonable summaries, it has potential limitations. Later, in 2010 Liu investigated meeting-specific features in order to improve the summarization quality. Such features as gender, speaker's role in meeting are normalized and incorporated into system. The experiments show that sentence length and position improve summarization quality more than normalized meeting-specific speaker features. A short time after, Liu et al. [LiXL10] investigated summarization based on n-best hypothesis selection and automatic keyword extraction. For hypothesis selection they used MMR [CaGo98] framework and TF*IDF for keyword extraction. The authors observe improvements in summarization and promising future directions.

2.1.2.1 Summariazation of Spoken Lectures

Rhetorical structures of lectures, broadcast news, and other types of discourse provide valuable information for extracting speech summaries automatically. A lecture follows a particular hierarchical structure. Typically, professors welcome the audience, introduce the topic and the background information, problems, methods, applications, and results. At the end, most lecturers conclude and summarize the lecture.

Zhang et al. [ZhCF10], [ZhHF08] proposed K-Means and Rhetorical-State Hidden Markov Model (RSHMM), two approaches to automatic summarization of spoken lectures, which utilize rhetorical structure of lectures. Both approaches consist of training and testing. During the training phase, presentation slides are segmented into three sections: introduction, content and conclusion. The section boundaries are defined based on cosine similarities between sentences in transcription.

During the test phase, the lecture is initially split into three sections. Each sentence is represented through a feature vector which spans acoustic, linguistic and discourse features. To reduce the dimensionality of feature vectors, Principle Component Analysis (PCA) is applied and feature vectors are projected into two dimensional space.

K-means algorithm follows PCA which clusters the sentences into different sections. This process is repeated until section boundaries remain unchanged. However, it is important to notice that during the training, presentation slides have to be available for the segmentation , but during the testing , it is assumed that the corresponding slides are missing. After segmentation, informative sentences are extracted from individual segments and combined, thereby building a summary.

The K-means approach considers only the separate segments, however, RSHMM also supports the transitions between the segments by using HMM. The Rhetorical State HMM approach considers R HMMs for R sections where each HMM has three states corresponding to beginning, middle and end of a section. During HMM training sentences are labelled with sections such as introduction, content, conclusion, etc. During the testing phase, the sentences of the test lecture are assigned to sections based on segmentation learned during training. After segmenting the lecture with RSHMM, a summary is selected from segmented test lectures using SVM binary

classifier. SVM classifier uses a probability threshold to decide whether the sentence is in summary or not.

Both K-Means and RSHMM use acoustic and linguistic features for segmenting lectures into rhetorical units. In their previous work in 2007 [ZhCF07] Zhang et al. show that segmentation works partially by using only linguistic features and does not work at all if only acoustic features are used. The best segmentation is achieved by using both acoustic and linguistic features. Acoustic features include time duration of the sentence, average syllable duration such as F0 I, F0 II, F0 III, F0 IV, F0 V, E I, E II, E III, E IV and EV. Since these features are speaker dependent, speaker-normalized acoustic features such as PF0 I, PF0 II, PF0 IV, PE I, PE II, PE IV are more accurate. Linguistic features, on the other hand, include the number of words in the current, next and previous sentences, the TF*IDF, TF, IDF scores, and the cosine similarity measure. Also, discourse features such as the probability distribution of nouns are considered.

The disadvantage to using either approach, K-Means or RSHMM is the assumption that a lecture's presentation slides are available for training, and this is not always the case. Furthermore, some linguistic features such as sentence length, do not indicate important information for summarization. Therefore, the use of these approaches is limited to those lectures which slides and grammatically and semantically correct sentences. Moreover, the authors assume that the slides of lectures are available and that human annotations are generated based on these slides. This assumption is not always true since sometimes lectures are held without slides.

Additionally, Zhang et al. investigated approaches for automatic lecture annotation, which allows to reduce the effort for creating reference summaries. Having only a small set of annotated lectures, active learning algorithm allows to predict the labels of transcribed lecture sentences and improves the production of gold standard summaries. In the paper in 2009 [ZhCF09], Zhang et al. applied active learning to align unlabelled lectures with PowerPoint presentations using a similarity score. In their latest work [ZhFu12] they proposed another approach based on active learning and Relaxed Dynamic Time Warping (RDTW) alignment between speech and its accompanied presentation slides.

Finally, the most recent approach to automatic lecture summarization, a procedure developed by Yun-Nung Chen et. al. [CHYL11] in 2011, is based on a random walk over a graph. This approach assumes that sentences similar to the most important sentences are more important. Similar to other graph-based approaches, a graph uses sentences as nodes and similarities between sentences as edges. The authors define similarity between sentences as those sentences which share topic words and phrases. To identify topic similarities, a graph is constructed using automatically extracted keywords. The keywords are then extracted using Probabilistic Latent Semantic Analysis(PLSA) and statistical measurements of terms. The experiments show that these keywords significantly improve the summarization quality. However, the main disadvantage to this approach is identifying the topic of the lecture. The authors assume that topics are defined in advance, which is often not the case for many lectures.

2.2 Segmentation

Similar to summarization, there are several different tasks and approaches in text segmentation. Most of these approaches are based on similarity measures and show comparable good performance. However, most segmentation approaches have been actively applied on videos and only few of them have been applied on text. Bellow we introduce some of those approaches in detail.

One of the earliest approaches in text segmentation was proposed by Hideki Kozima in 1996 [Kozi96]. Hideki proposed a new structure indicator called lexical cohesion profile(LCP) which allows to detect segment boundaries in text. LPC detects the similarities between words in sentences. The similarity is computed based on semantic network. LPC is also very useful for resolving anaphora and ellipsis. The authors show that the applying LDC leads to reasonable results in segmentation, anaphora and ellipsis resolution and propose to integrate cue words in the future.

One of the most popular segmentation tasks is topic segmentation. One of the earliest approaches in topic segmentation was proposed by Ponte et al. [PoCr97] in 1997. Their work uses Latent Context Analysis (LCA) to detect sentence related words and phrases. Sentence related words and phrases are compared pairwise in order to find similarities and score the segments in various sizes. Ultimately, dynamic programming is applied to segment the text. Experimental evaluation of proposed approach showed that LCA works well for short passages but the length of the passage needs to be optimized.

Another approach in topic segmentation was introduced by Shafiei et al. [ShMi08] in 2008. The authors introduced hierarchical Bayesian structure for identifying topic related segments and assigning them to different topics. It uses hierarchical structure to find the correlations between words and topics. Proposed system showed comparable performance with other segmentation approaches.

Later, In 2000 Rainer Linenhart and Wolfgang Effelsberg proposed and approach to automatic text segmentation for video indexing using unique features [LiEf00]. The unique feature in this approach is the tracking of terms over their complete time of occurrence in video. The extracted text from videos is later shown on binary images. The authors show high performance of text segmentation.

A further approach to text segmentation used for segmenting movie subtitles was proposed by Martin Scaiano et al in 2000 [SILR10]. The authors applied different modifications of a well-known segmentation approach, TextTiling on movie subtitles. For example one modification was the enhancement of cosine similarity by Word-Net based method. They also considered a synset instead of vector of words with cosine similarities which resolved the disambiguation. However the authors mentioned in their work that their approach shows effective results only for subtitles of movies and is not necessarily effective for other domains.

One of the latest approaches in video segmentation was proposed by Balagopalan A. et al. [BBBC$^+$] in 2012 where the authors applied Naive-Bayes classifiers for automatic keyword extraction and segmentation. Naive-Bayes allows to classify the terms in *Keyword* and *Non-Keyword* classes. In the training phase the approach uses feature vectors representing video lectures to learn statistical model. Then the model is applied on unseen video lectures to extract the keywords. Although

the approach shows reasonable segmentation results for video lectures it performs differently when applied on Automatic lecture recognition output. This is because automatically recognized speech deals with different type of problem such as sentence boundaries and speech disfluency.

3. Challenges in Lecture Segmentation and Summarization

Spoken communication is the easiest mean for exchanging information or teaching people about different subjects. That's why educational institutions all over the world offer lectures as a main resource for teaching students about particular subject. There are different teaching methodologies but all of them aim that the audience understand the subject being discussed. In order to make their talk understandable the educators follow particular structure. They begin from a simple introduction and evaluate the subject deeper in preceding paragraphs. Additionally, body language, voice changes, speech disfluency, intonation and interaction with the audience play a significant role in the comprehensibility of lectures. In this chapter we discuss different characteristics and challenges of spoken lectures. We discuss different structural characteristics and challenges such as Multi-channel interaction, speech disfluency dialect and ascent with the help of examples.

3.1 Structure dependency

Many lectures can be characterized through particular hierarchic structure. The structure helps us to understand the subject easier, starting from very simple examples and getting gradually more and more complex.

The structure usually varies from lecture to lecture, however, there are some structural units which are present in almost all lectures. The professors structure the lectured based on their preferences. Therefore, lecture structuring can be very subjective. However, such section as introduction, content, asking, answering questions and conclusion are common in almost every lecture.

Less common but important structural units are revision, background information, examples, methods, experiments and discussion. Some topics can be discussed over many lectures. Therefore, many professors used to revise the subject from previous lectures to remind students about the topic they are going to continue. In this

case background information section takes the place of revision. Usually if there is no revision professors introduce background information. By further development of the topic professors explain the core ideas and often use examples for better understanding. Short before conclusion they talk about experiments which are of course topic specific. At the very end depending on time limitations they have sometimes discussion.

The structural information of lectures can be used to segment the lecture into paragraphs and improve the quality of summarization. Segmentation will allow us to focus on content-relevant structural units and discard or penalize content-irrelevant units in summarization phase.

3.2 Paragraph and Sentence Boundaries

One of the major challenges in lecture segmentation is the detection of boundaries both for sentences and segments. Lecture transcriptions, which are automatically generated by Automatic Speech Recognizer (ASR) contain no paragraphs, sentences are often incomplete or have missing boundaries. This makes whole lecture hard to read and leads to problems by understanding the subject. In contrary to sentence boundaries, which are more or less objectively clear, segment boundary identification is very subjective, sensitive and has low inter-annotator agreement. Bellow we can see a part of an example transcript

this is not true thus , I will not read but this here contained several glosses anyway , looking at it was obviously meant so and I think we have a good appreciation for handling also the humorous level

3.3 Multi-Channel Interaction

Many educators arrange lectures interactively through discussions and question-answering dialogues. Both questions and discussions often relate to the lecture but there are some exceptions too. For example a discussion can be started or an announcement can be done about logistic information such as the schedule of the classes, location, final exams and so on. One of the big challenges in multi-channel interaction is the interruption. Professor's speech can be interrupted with questions or abruptly a discussion can be started. In this case if we follow the lecture we need to detect the position where the speech was interrupted and continued further.

Interaction presents a further difficulty for automatic speech segmentation and summarization. To make the summaries more sophisticated it is important to detect the boundaries of interactions. This will help to remove content irrelevant parts from the summary.

3.4 Speech Discourse

Speech discourse means verbal communication or conversation and is a challenging topic for linguists. Discourse analysis examines style, rhetoric, speech act and use of prepositions. Different type of discourses has been examined in politics, education, health care and media. Speech discourse in education concerns to speech of educator or presenter.

In universities, speech discourse varies from lecturer to lecturer and depending on speaker leads to different speech recognition qualities. Despite certain commonalities, each person has his individual style of rhetoric which causes to different levels of clearness and understandability of the speech. Different discourses lead to different speech recognition qualities and this has a huge impact on post processing of the recognized text such as segmentation and summarization. The better the quality of speech recognition, the more understandable is the spoken language.

3.4.1 Dialect and Accent

Dialects are the varieties of a language which extend the language with region-specific vocabulary and grammar. They are associated with regional groups and are difficult to understand for non-regional people.

In contrary to dialect, accent relates only to pronunciation of speech. In this case we assume that the sentences are grammatically correct and the vocabulary stays unchanged.

Both dialect and accent effect the quality of speech recognition thereby having a negative influence on automatic speech summarization. They cause misspelled vocals, vocabulary extensions and non-grammatical sentences which are very critical during speech recognition. Vocabulary extensions and speech patterns of dialects are unfamiliar to speech recognizer. This leads to words out of vocabulary and miss-recognition. Ultimately, the summary of miss-recognized text becomes error-prone and miss-understandable.

3.4.2 Disfluency

Speech disfluency decreases the readability and conciseness of automatically generated summaries. It implicates false starts, full-words, non-lexical and misspelled words which make speech inarticulate and lengthy. This leads to the text which is poorly understandable after speech recognition. The example bellow shows the effect of disfluency.
Hmmm... So, so welcome to today's lecture hmm ...
well we have two things on our our agenda we will talk dynamic logic
well let's start
However, there are many different methods to post-process speech-to-text output further and clean it up from disfluency. For example, we can use a dictionary to detect Out Of Vocabulary (OOV) words and part of speech tagging to detect filled pauses(uh, ahm, um, ...) and repetitions as mentioned in Zecher's and Waibel's work [ZeWa00]. However, sometimes speech disfluensy is delusive and leads to ambiguity. This means that speech disfluency detection can become a very challenging task.

Intonations stress and timing patterns can be both very helpful and misleading in detecting the boundaries. If the intonations stress and timing patterns are misplaced due to disfluency they can lead to wrong boundary. In meanwhile if they are used correctly they can be very helpful in detecting sentence and segment boundaries.

3.5 Automatic Speech Recognition Problems

Another type of errors are the errors caused by Automatic Speech Recognizer (ASR). This type of errors are especially excessive in spontaneous speech, such as in lectures where the recognition accuracy is a little bit worse than 70%. This has to do with the less constrained nature of spontaneous speech. Also the differences in use articulation, environmental noise and microphone play significant role in the ASR performance. Sometimes ASRs are performing very well for some users and poor for others. This has to do with speaker adaptation. For example some ASRs are well adapted for some speakers and deliver good recognition results, whereas they perform worse for unfamiliar speakers with significantly different articulation, for example. Although many systems have been developed to overcome these problems, there are no perfect solutions which cover all possible cases. Existing ASR error correction software usually does post-processing on the ASR output based on statistical methods and have a significant impact on the quality of recognized speech.

3.6 Conclusion

Spoken-lectures possess many different speech-specific features and challenges which are not present in written text. Many of these challenges increases error-proneness of automatic speech summarization. Such challenges as speech disfluency, dialect, accent, wrong sentence or segment boundaries and multi-channel interactions have a negative effect on readability of summary. However, structure-dependent property of lectures is used to improve the quality of summarization. Since most of lectures follow a particular structure, we can use this information in summarization process.

4. Spoken Lecture Segmentation and Summarization

4.1 Goal and Requirements

The goal of this thesis is to develop a segmentation and summarization system for spoken lectures. The system receives spoken lectures in form of text as an input, segments them in structural units and then summarizes those units by extracting content-relevant and salient information. The outputs of our system are structured lectures with paragraphs and its concise, human-readable summary.

The requirements of this thesis are to break up lectures into paragraphs and generate long and short summaries by extracting most important information. The paragraphs makes lectures readable and the summaries condense the content by removing irrelevant information. The system faces several limitations. One of the limitations are lecture slides. Since the slides are available only for few lectures they cannot be used as a resource during summarization. Another limitation is the quality of automatically recognized lectures. Poor quality of automatically recognized lectures leads to summaries of poor quality. That's why we use manual transcripts for the evaluation of our system. To evaluate the system we used English lecture transcripts, however the system supports also other languages and is developed to work as language independent.

4.2 System Architecture and Overview

The overall system architecture is depicted in Figure 4.1. The input of our system are lecture transcriptions which are cleaned from speech disfluency and redundant information. We also assume that sentence boundaries are inserted in input transcriptions. In this paper, we consider the sequence of cleaned transcription sentences as the input to the segmentation/summarization system. The final system output is a concise and formatted summary with proper paragraph boundaries.

The system consists of five main components: *pre-processor*, *feature extractor*, *segmenter*, *summarizer* and *merger*.

- *Pre-Processor* eliminates the noise and tokenizes the text into comprehensible units. For example, such constructions as "I'm", "today's" are dropped into such units as "I", "'m" and "today", "'s". Also we eliminate all information about case and incomplete words.

- *Feature extractor* is responsible for extracting various structural, lexical, contextual features on sentence level by representing sentences through its feature vector. Extracted features capture for example sentence position, sentence length, TF-IDF score, sentence similarities and some other features described in Sections 4.4.2 and 4.3.2. Overall we differentiate between segmentation and summarization features. Some of segmentation features are used also in summarization, such as "sentence relative position" feature.

- *Segmenter* segments the input sentence sequence into a hierarchical structure such as "introduction", "content" ("content-topic$_1$", "content-topic$_2$", ... , "content-topic$_n$"), "conclusion".

- *Summarizer* summarizes each segment of the lecture using summarization features including *TF-IDF score*, *sentence similarity*, *centrality* and other features described in Section 4.4.2. Each segment uses specific summarization models that depend on the rhetorical function of the segment.

- *Merger* receives the summarized segments from summarizer and assembles them together thereby building the entire summary. During this process the merger considers predefined sequence of segments. For example, summaries begin with introduction segment, followed by background information and end with conclusion segment.

For training proposes the system uses additional data sources as shown on Figure 4.1. Lecture training corpus, scientific papers and TED, a global set of conferences including talks in various subjects, are the main data sources used in our system. Lecture training corpus includes around 60 lectures, scientific papers are used only from segmentation module and include around 3000+ publications and finally, TED corpus includes around 1100+ talks.

4.3 Automatic Lecture Segmentation

Many lectures follow particular hierarchical structure with some commonalities between them. Common structural elements are introduction, body, question and conclusion. Less common structural units include revision, background, method, evaluation, logistic information. Figure 4.2 depicts an example of such structure. In this example we can see that such segments as method, evaluation, logistic information are missing. These segments are the sub-segments of content and are considered to be lecture specific.

Segmentation allows to detect and penalize content irrelevant segments of lectures thereby improving the quality of summarization. Such segments as questions, logistic information or introduction are less content relevant than background information or body and conclusion. This allows to prioritize the segments relating to summarization and consider some segments more important than the others.

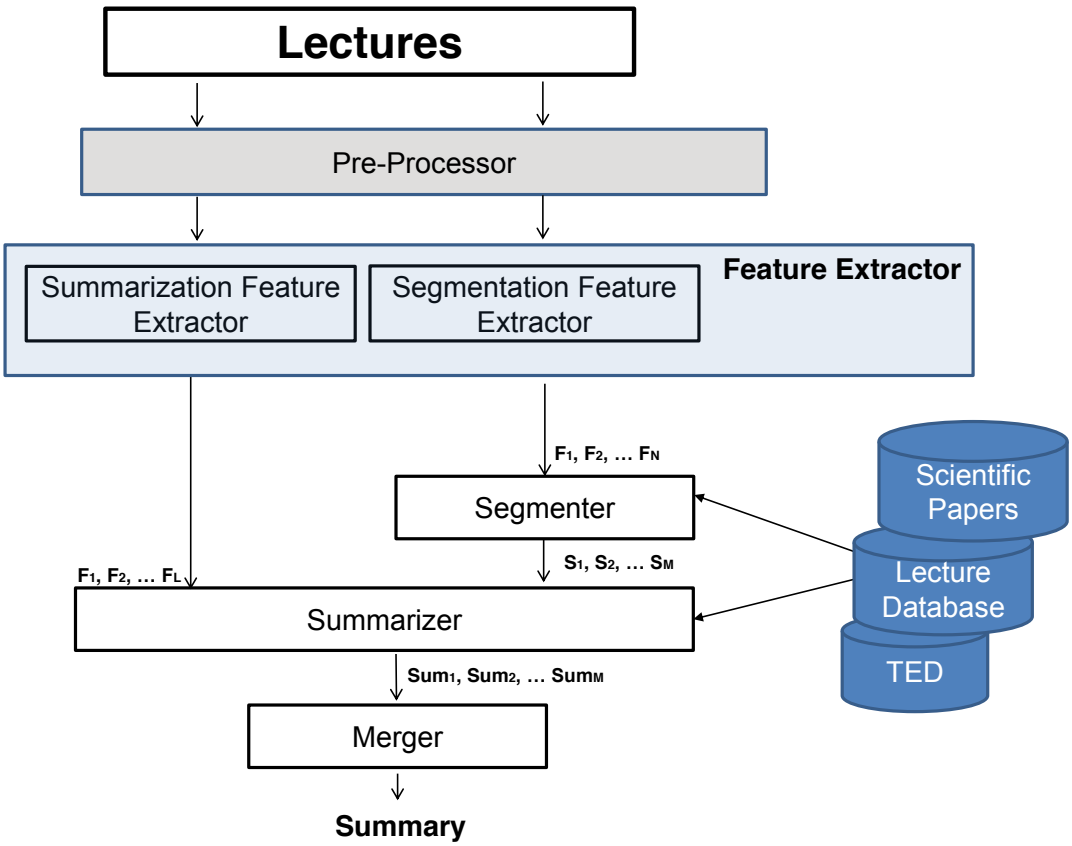

Figure 4.1: Overall System Architecture

We implemented three different approaches for automatic segmentation, namely *K-means cluster*, *hierarchical top-down segmentation* and *Hidden Markov Model (HMM)*. These approaches use the same set of features but different algorithms for segmentation.

4.3.1 Segmentation Methods

In this section we introduce three segmentation methods(HMM, K-Means and Hierarchical Segmentation) and a new metric, Rhetorical Structure Index(RSI). RSI estimates the connection of words or phrases to different structural units thereby assisting to sentence classification in segments.

4.3.1.1 Hidden Markov Model

First segmentation algorithm, HMM, represents each rhetorical segment in lectures as a state in the Markov model and the transitions between the states are transitions between the segments. HMM states emit the probabilities of each sentence feature vectors in that state. The sentence feature vectors are the representation of sentences though features such as sentence relative position, RSI score and content similarity between sentence pairs. Figure 4.3 depicts five HMM states and the transitions between those states.

HMM uses Expectation Maximization(EM) algorithm to learn HMM [aYou07] parameters from unlabelled lecture data. It uses EM based algorithm, Baum-Welch, to train statistical model and learns unknown segmentation parameters. The unknown

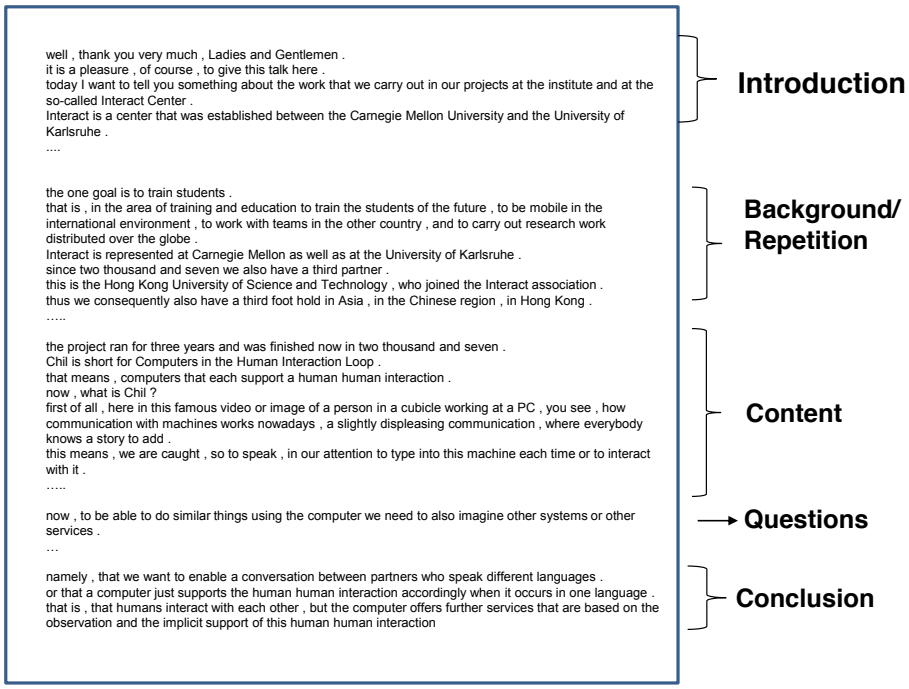

Figure 4.2: An Example of Lecture Segmentation

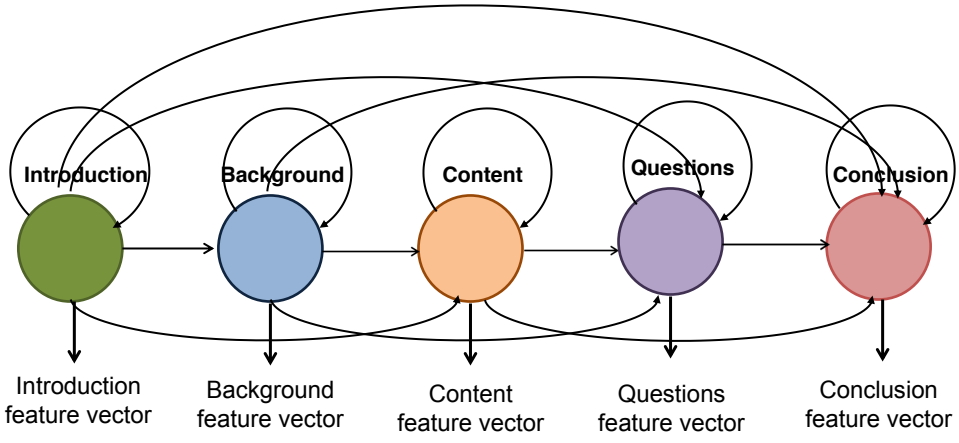

Figure 4.3: The representation of five segments(Introduction, Background, Content, Questions, Conclusion) and the transitions between those segments through HMM

parameters are emission and transmission probabilities which are estimated during training stage using backward-forward algorithm. Backward-forward algorithm estimates the probability of a sequence by combining forward and backward probabilities. Forward algorithm uses induction to calculate total probability of state i at time t. If we assume that $o_1, ..., o_{t-1}$ are the output observations till the time point t then the total probability at time t looks like the following:

$$\alpha_t(i) = P(o_1 o_2 ... o_{t-1}, X_t = s_i|\theta), 1 \leq i \leq N \tag{4.1}$$

It can be calculated using induction theorem. The basis case of induction looks like the following:

$$\alpha_1(i) = \pi_i b_i(o_1) \tag{4.2}$$

where b_i is the emission probability at state s_i. The induction step looks like the following:

$$\alpha_{t+1}(i) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_i(o_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N \tag{4.3}$$

where a_{ij} is transition probability. Similar to forward algorithm backward algorithm uses induction method too. In contrary to forward algorithm it starts from the end of the sequence.

$$\beta_t(i) = P(o_t o_{t+1} ... o_T | X_t = i, \theta) \tag{4.4}$$

After calculating backward and forward probabilities we combine them using the following equation:

$$P(O|\theta) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i), 1 \leq t \leq T \tag{4.5}$$

To segment unseen lectures, we apply Viterbi algorithm. Viterbi algorithm finds the most likely sequence of sentences from all segments for summary. It uses the emission and transition parameters learned during training phase and is defined as following:

$$i_1 = argmax_j(\pi(j) b_{jk_1}) \tag{4.6}$$

$$i_t = argmax_j(a_{i_{t-1}k_t} b_{jk_t}) \tag{4.7}$$

Initial HMM parameters(emission and transition probabilities) are bootstrapped using results from K-Means clustering.

4.3.1.2 K-Means Algorithm

One of the most intuitive approaches to segmentation is K-Means. In our set up, K corresponds to number of segments in desire for a lecture. Similarities are computed based on the content similarity (using n-gram matches) and the relative sentence position defined as:

$$\begin{aligned} Sim(S_i, C_j) = \alpha \cdot M(S_i, C_j) + \\ \beta \cdot P(S_i, C_j) + (1 - \alpha - \beta) \cdot R(S_i, C_j) \end{aligned} \tag{4.8}$$

where S_i is the i-th sentence, C_j is the centroid of the j-th cluster, $\cdot$ is the dot product between S_i sentence and C_j centroid, $M(S_i, C_j)$ is the content similarities between

sentence S_i and centroid C_j. Similarly, $P(S_i, C_j)$ for sentence-position-in-lecture's similarities. $R(S_i, C_j)$ is similar to $M(S_i, C_j)$ but weighting the matched word/n-grams using their corresponding Rhetorical Structural Index (RSI) values described in more details in Section 4.3.2.1. High value of a weighting factor means that similarity measuring component connected to that weight is more trusted. Position similarity is based on the relative position distance between the sentence and the cluster:

$$P(S_i, C_j) = \frac{|D|}{Pos(S_i) - Pos(C_j) + \epsilon} \tag{4.9}$$

where S_i is the i-th sentence, C_j is the j-th cluster, $Pos(S_i), Pos(C_j)$ are sentence and cluster relative positions and $|D|$ is the total number of sentences in document. ϵ is a small constant to avoid divide by zeros.

Content similarity is based on the number of word matches between a sentence and a cluster center. Denote the binary word frequency vector (bag of words) in sentence S_i as $\vec{S_i}$ and similarly $\vec{C_j}$ for cluster centroid C_j. The more matches the sentences have the similar they are. The similarity between a sentence and a cluster centroid is denoted by:

$$M(S_i, C_j) = \frac{\vec{S_i} \cdot \vec{C_j}}{\| \vec{S_i} \| \| \vec{C_j} \|} \tag{4.10}$$

Let's compute the similarity on an example of three word frequency vectors(see Figure 4.4). It is easy to observe that the first and second vectors have two matches, first and third only one match, second and third one no matches. According to equation 4.10 the similarities between first and second, first and third binary vectors are 1, second and third 0.

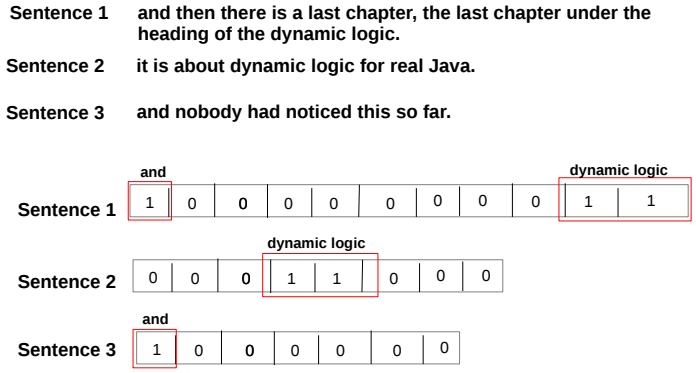

Figure 4.4: An example of binary word frequency vectors(bag of words) where the boxes marked in red denote the similarities.

4.3.1.3 Hierarchical Segmentation

Inspired by the work of [WPZZ11] where time-series sensor input is segmented into a hierarchy of human activities, we apply the same algorithm on the time-series lecture data, in this case, a sequence of sentences of lecture transcription. Similar

to [WPZZ11], we apply a sliding window of size W over each position t of the lecture and calculate the difference between contents inside window $(t - W, t)$ and $(t + 1, t + W)$. Figure 4.5 shows an example of topic changes in segments for window sizes 5, 20 and 50. A "peak" at the curve of "changes" with respect to t indicates that at position t, the topic of the lecture is likely changed at that point. To identify the peaks we observe continuous divergence of sentences from topic in an interval. If the divergence is abruptly at a one point t then it is less likely that t is the time point when topic changed. Figure 4.6 shows two examples of topic changes. The first example on the left side shows continuous divergence of sentence from topic content and arises a peak in time t. The second example on the right side shows an abrupt(discrete), big change of a sentence from the topic and therefore does not considered to be a peak.

With a large W, we identifies positions where high-level topic changes occur and segment the lecture at each positions. Within each segments, we continue the above process with a smaller W to further locate "changes" with finer granularity and repeat this process which results in a hierarchical segmentation of the whole lecture. The advantage of this approach is that we can always hierarchically go into segments and make the paragraphs as fine granular as we wish. Figure 4.7 shows hierarchical segmentation on a lecture example.

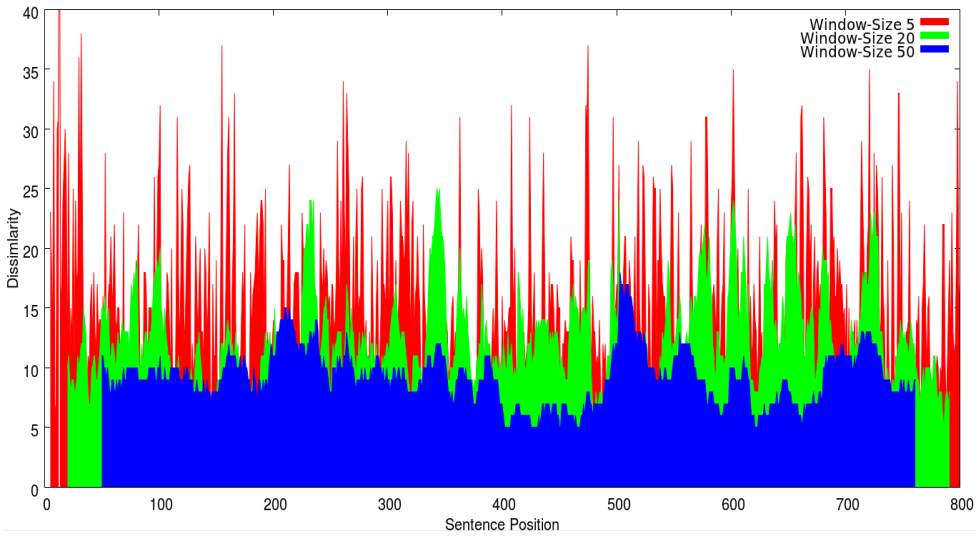

Figure 4.5: Topic changes in segments using three different window sizes. X-axis is the position of sentences in a lecture and Y-axis is the "change" of topics between two adjacent window $(t - W, t)$ and $(t + 1, t + W)$. Shown in this figure are three curves each correspond to a different window size which in turn captures the "topic change" at different granularities.

4.3.2 Segmentation Features

Segmentation features allow to detect particular structural characteristics in segments. These characteristics can be used to separate lectures in different segments.

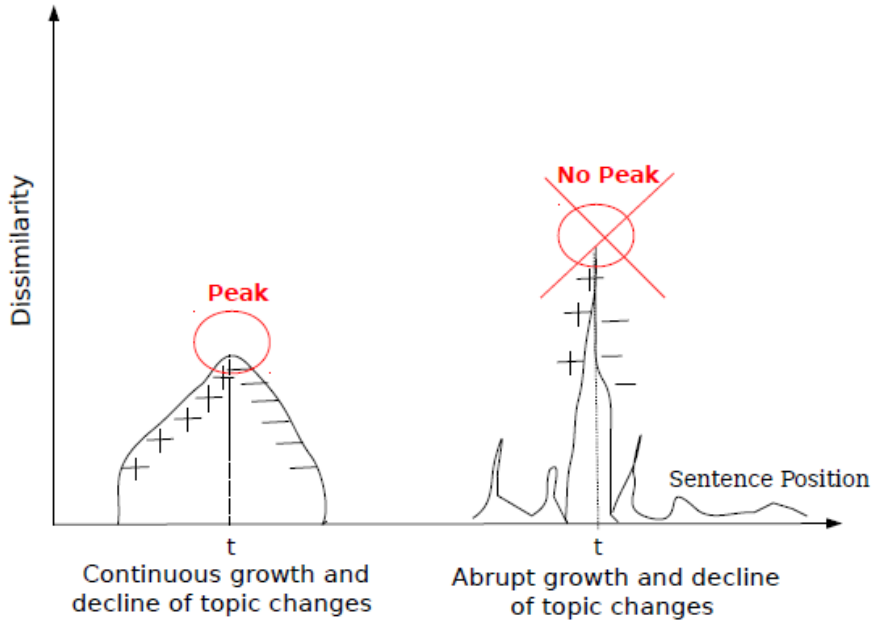

Figure 4.6: Continuous topic changes emerge a peak(on the left side) and abrupt big changes don't(on the right side).

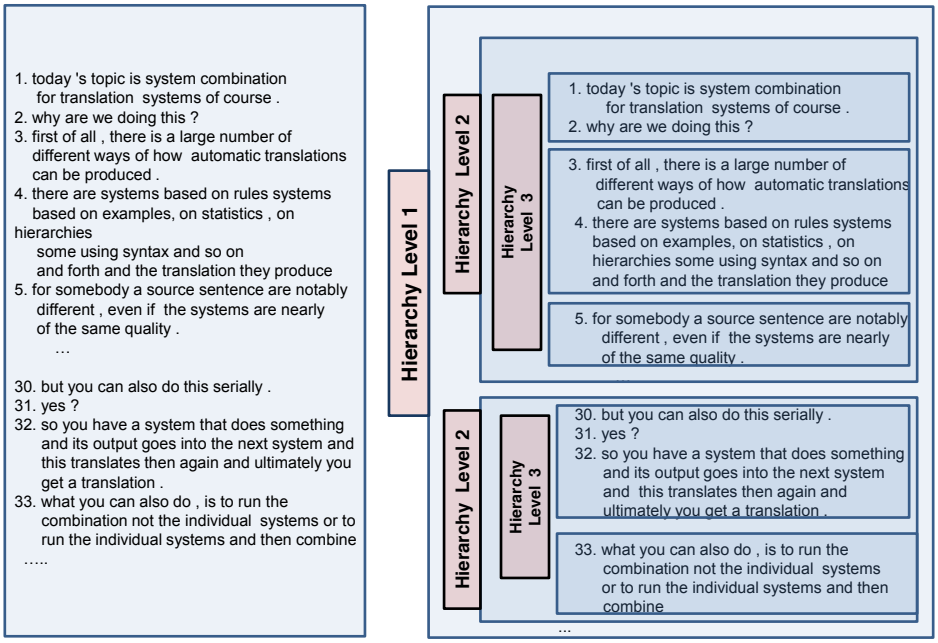

Figure 4.7: Three level hierarchical segmentation on an example lecture. On the left side we can see the original lecture before segmentation and on the right side after segmentation

In this chapter we discuss different segmentation features. One of these features is the distribution of words. Words with a small variance of distribution tend to be connected to a particular segment of lecture. These words can be good indicators of segments, if we consider a large number of documents. The number of lectures used for detecting the variance of distribution reaches 70 and the number of papers 3000+. We detect those indicators of segments using RSI. RSI is described detailed in chapter 4.3.2.1. A further feature of segmentation is the similarity of sentences. Typically, similar sentences tend to belong same segment. The similarity of sentences is computed based on similarity of content words. The more the content words overlap, the similar the sentences are. For example these two sentences are more similar

and then there is a last chapter, the last chapter under the heading of the dynamic logic.
it is about dynamic logic for real Java.

due to two important content words than these two:

and then there is a last chapter, the last chapter under the heading of the dynamic logic.
this was important for the conflict resolution for quantified updates.

Since they have only one word in common.

Another obvious segmentation feature is the position of sentences. Adjacent sentences are more likely to be in same segment than the distanced ones.

4.3.2.1 Rhetorical Structural Index

Segmentation is a different task than information retrieval or summarization. In segmentation, we want to find the rhetorical structure of a document or a lecture. Intuitively, phrases such as "the topic of today's lecture is", "the outline is", "now let's switch gear to", "next, I will talk about", "any questions", "in conclusion, this talk" are good indicators of the structure of the lecture even though their TF-IDF scores may not be very high as they do not bear too much information content-wise. For segmentation task, we propose a new metric called Rhetorical Structure Index (RSI) which aims to assign weights to words/phrases based on their *structural importance*. RSI considers both frequency of occurrence and more importantly, the variance of distribution. The intuition is that if a word or phrase is a structural marker, it usually occurs at a certain position of a document or lecture.

We define RSI as:

$$RSI(w_j) = \lambda_1 \cdot \sigma(\mathrm{P}os(w_i)) + \lambda_2 \cdot \frac{|D|}{|\{d \in D | w_i \in d\}|} \tag{4.11}$$

where $\sigma(\mathrm{P}os(w_i))$ is the variance of w_i's relative positions defined as $Pos(w_i)$, $|D|$ is the number of all documents, $|\{d \in D | w_i \in d\}|$ is the number of documents where w_i appears, λ_1 and λ_2 are the weighting factors, $\lambda_1 + \lambda_2 = 1$. Tables 4.1 and 4.4 show the RSI values for some n-grams from the lecture transcription. Table 4.2 and 4.3 show the RSI values for some n-grams using statistics calculated from ACL conference papers. The lower the RSI value, the more structurally important a word/phrase is.

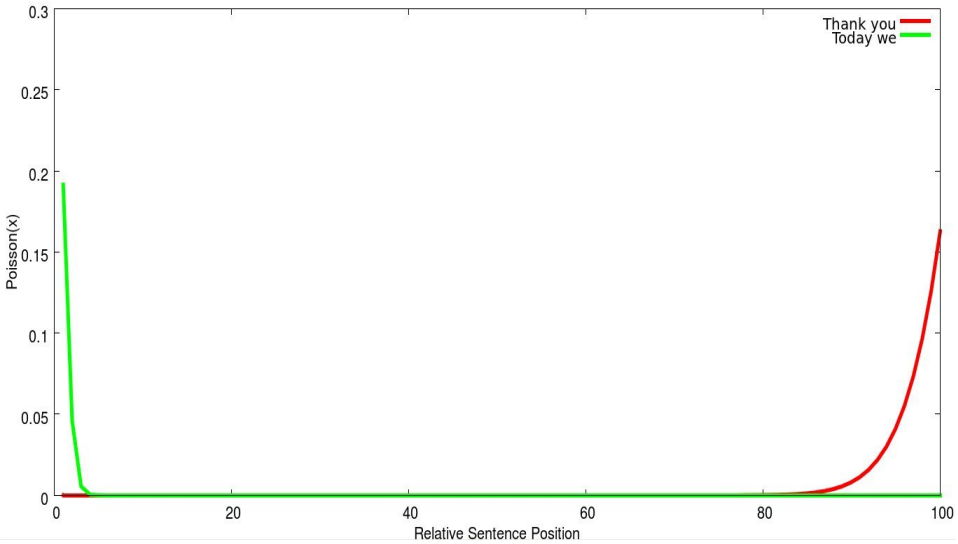

Figure 4.8: Word distibution of two example words, *"today we" and "thank you"*. *"today we"* has dense distribution in introduction segment, whereas *"thank you"* in conclusion. X-Axis of the diagram shows relative word position in whole document and Y-Axis shows the probability.

It is worth noting that positions of structurally important words/phrases turn to follow a Poisson-Mixture distribution described by equation 4.12.

$$p(x, \lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!}, x = 0, 1, 2, ... \tag{4.12}$$

where λ is the expected value or variance of x and characterizes the shape of the distribution.

Figure 4.8 shows the distribution of two phrases in lectures: *"today we"* in introduction and *"thank you"* in conclusion segments.

Detected keywords are stored in separate data files and used later as segmentation or summarization features.

4.4 Automatic Structure-dependent Lecture Summarization

4.4.1 Summarization methods

To identify the most appropriate approach for lecture summarization we investigated different possible directions. The first direction was word selection. The approaches based on word selection select important, cue words and phases by using different metrics and combines them together thereby building a new summary. The second direction was sentence extraction. The approaches pursuing sentence extraction, select salient sentences based on different features or metrics. Extracted sentences

Table 4.1: Rhetorical Structure Index(RSI) values of unigram, bigram "introduction" and "conclusion" of lectures. The lower the RSI value, a word/phrase is structurally more important.

Unigram				**Bigram**			
Introduction	RSI	Conclusion	RSI	Introduction	RSI	Conclusion	RSI
week	0.016	sense	0.023	dealing with	0.017	such that	0.024
class	0.022	application	0.048	see how	0.0238	next time	0.111
step	0.031	ultimately	0.048	regarding the	0.0476	thank you	0.115
tell	0.033	doing	0.056	talk about	0.069	case of	0.204
dealing	0.049	somehow	0.056	today we	0.071	needs to	0.231
higher	0.051	human	0.062	front of	0.074	tell you	0.238
belongs	0.063	certainly	0.062	ladies and	0.105	means that	0.257

Table 4.2: Unigram, bigram keywords and Rhetorical Structure(RSI) Index in scientific paper introduction and conclusion segments.

Unigram				**Bigram**			
Introduction	RSI	Conclusion	RSI	Introduction	RSI	Conclusion	RSI
years	0.728	supported	0.432	using a	0.600	models of	0.168
recent	0.954	future	0.854	used in	0.680	analysis of	0.199
here	0.987	available	0.931	one of	0.948	introduction to	0.208
current	1.070	described	1.081	propose a	0.976	algorithm for	0.228
presents	1.121	thank	1.084	task of	1.019	study of	0.230
particular	1.163	presented	1.145	order to	1.046	approach to	0.237
useful	1.178	first	1.193	present a	1.058	based on	0.258

Table 4.3: Rhetorical Structure Index(RSI) of trigrams in scientific paper introduction and conclusion segments.

Trigram			
Introduction	RSI	Conclusion	RSI
paper we focus	0.162	work was supported	0.081
used in the	0.162	supported in part	0.102
shows that the	0.173	future work we	0.110
show that it	0.186	presented a novel	0.115
experiments show that	0.191	paper we presented	0.121
applied to the	0.191	partially supported by	0.121
paper we introduce	0.197	show that our	0.132

Table 4.4: Trigram keywords and Rhetorical Structure(RSI) in lecture introduction and conclusion segments.

Trigram			
Introduction	RSI	Conclusion	RSI
last time we	0.021	next time we	0.028
here we have	0.024	time we will	0.028
look at the	0.062	here you see	0.104
ladies and gentlemen	0.096	thank you very	0.119
needs to be	0.167	here we have	0.233
first of all	0.167	look at the	0.357

are combined together and used as a summary. In order to find out the most human preferable approach, we manually summarized a lecture using both word and sentence extraction and created a survey using those summaries. The survey asked humans to choose the best summary of a paragraph by selecting either word, sentence extraction or 'no significant difference'. Bellow we can see an example paragraph of word selection summary and an example of sentence selection summary bounded correspondingly by solid and dashed boxes.

welcome to lecture we have two things.
firstly I want to finish the subject matter of lecture did not finish the updates.
after we had treated updates we described the semantic, which was not that easy.
I was astonished that one of you has found a mistake on the slide. the corrected slides are already on the server.

we have two things on our agenda.
firstly, I want to finish the subject matter of our last lecture. we did not totally finish the updates yet.
let us go back in order to the last lecture have a look what it was about.
after we had treated updates we described the semantic which was not that easy one of you has found a mistake on the slide.
I have already shown the slides several times. the corrected slides are on the server the error is already corrected .

The evaluation results of survey showed that humans prefer sentence extraction versus word selection. On Figure 4.9 we can see the results of the survey based on 16 attendees. Blue part of the pie-chart shows the number of attendees who preferred sentence extraction, red part the number of attendees who prefers word extraction and green part the number of attendees who saw no significant difference.

4.4.1.1 Word Selection based Summarizer

Since existing open source summarization programs mostly implement sentence extraction, we developed a word selection based approach to compare with the existing ones. The algorithm is based on the approach proposed by Chiori et. al.(2003) [HFMY+02] and was evaluated using training data of different genre and size. Our implementation merges multiple sentences by selecting words and combining them together into one sentence. Word selection is based on the linear combination of

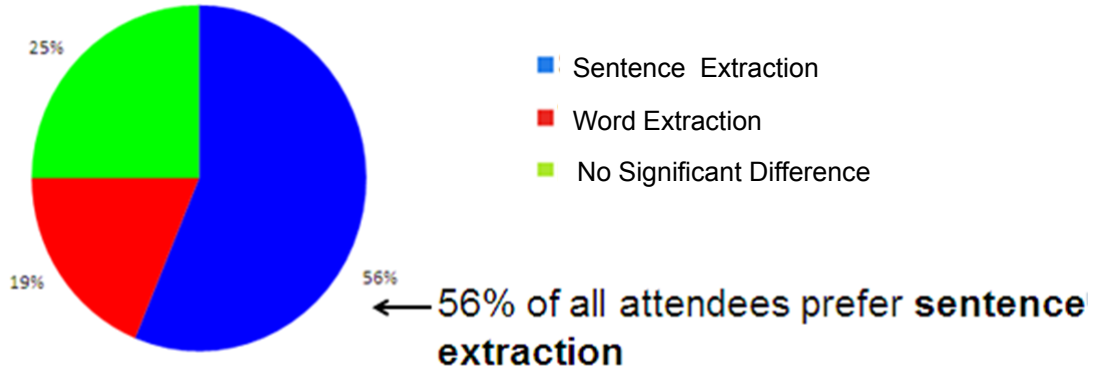

Figure 4.9: The results of human survey for evaluating the best summary based on 16 attendees. The colors indicate different possible options: "Sentence Extraction", "Word Extraction" or "No Significant Difference".

two different scores. The first score, so called linguistic score, indicates tri-gram language model score. For example, if w_n is current word, tri-gram probability will be signified by $P(w_n|w_n - 1, w_{n-2})$. The second score denotes the significance of the words and similar to TF*IDF metric considers word frequency.

$$I(w_n) = f_n * log\frac{F_A}{F_n} \tag{4.13}$$

where f_n is the frequency of w_n in a spoken lecture, F_n indicates the number of occurrences of w_n in all training data and F_A is the sum of all $F_n(F_A = \sum_i F_n)$ The authors proposed two more scores, concatenation and confidence scores, in their paper. However, according to the experimental results pointed in Chiori's work [HFMY+02] those two scores had only a little impact on summarization performance. The linear combination of above mentioned scores is called summarization score.

Let's assume that D is the summary sentence containing M words $D = w_1, ...w_M$. The algorithm selects the words which maximize summarization score given by:

$$S(D) = \sum_{m=1}^{M} P(w_m|w_{m-1}, w_{m-2}) + \lambda_I I(w_m) \tag{4.14}$$

where λ_I is a weighting factor for significance score. Dynamic programming allows to calculate the word sequence with maximum summarization score. Figure 4.10 shows an example of dynamic programming for extracting a sentence of 14 words out of five sentences.

4.4.1.2 Sentence Extraction based Summarizer

Summarization module receives the output of segmentation module and emits the summary of each module. It processes further the output of segmentation module by selecting important and content-relevant sentences. To select important sentences we examined two different approaches. The first approach, so called Support Vector Machine(SVM) [Vapn95] is a classification approach based on supervised machine

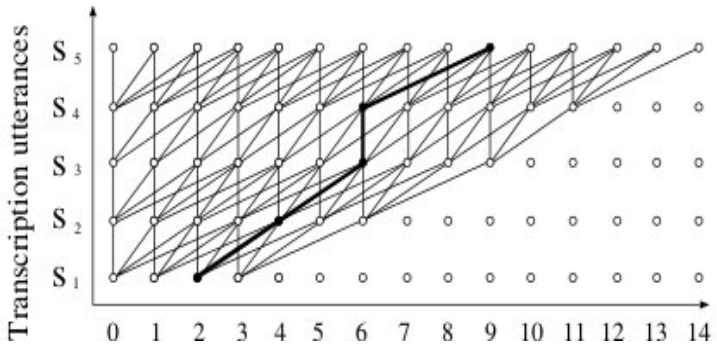

Figure 4.10: An Example of dynamic programming using multiple sentences

learning and the second approach, Conditional Random Field(CRF) [LaMP01], is a statistical model for labelling the sequences.
SVM allows to classify the sentences into "summary" and "non-summary" binary classes using statistical model trained on a feature set. It uses Radial Basis Function(RBF) kernel and lexical, contextual features to train the model. All features used in SVM are described in chapter 4.4.2.
Similar to SVM, CRF bases of different features and assigns "summary" or "non-summary" binary labels to each sentence. The advantage of CRF over SVM is that CRF not only considers the individual features of sentences but also their sequences. Furthermore, CRF allows to use a large number of features which is not the case for SVM.

Similar to other machine learning approaches, both SVM and CRF consists of two phases: training and test. During training phase the system learns model parameters based on manually annotated training data using different summarization specific features described in 4.4.2 section. The training data comes from lectures domain and consists of 4 lectures with the total duration of 209.97 minutes. During test phase learned model is applied on test lectures to summarize unseen lectures.

The number of sentences selected from each segment depend on the importance of segment. Let's assume summarization ratio is R, the number of sentences in lecture is N and the number of sentences in summary is S_N. Then, the number of sentences is calculated by the equation: $S_N = R * N$ and the proportion of sentences selected from each segment will correspond to:

Introduction $\rightarrow 0.01 * N$
Body $\rightarrow R - 0.03 * N$
- Methods $\rightarrow \frac{0.7}{R} * N$
- Evaluation $\rightarrow S_N - (\frac{0.7}{R} * N + 0.03 * N)$
- Questions $\rightarrow 0$
- Logistic Information $\rightarrow 0$

Conclusion $\rightarrow 0.02 * N$

We've predefined the proportion of sentences in each segment. For example, introduction comprises 10% of all sentences since the first introductory sentences are less

content relevant. In contrary to introduction, conclusion comprises 20% of all sentences since conclusions usually summarize key ideas of a lecture and are important. Methods comprise 70% of whole lecture. The rest is the evaluation results and experiments. We exclude Questions and Logistic information from the summary that's why these both segments comprise 0%.

4.4.2 Summarization Features

Summarization features help to identify content related sentences. The combination of these features allows us to score the sentences and pick up the ones with highest score. This chapter describes those features in detail.

Term frequency inverse term frequency(tf-idf), a known measurement metric of informativeness in information retrieval, assigns each sentence a score based on the weights of all terms in sentence and is defined by equations 2.1, 2.2 and 2.3. The sentences with highest TF-IDF score are included in summary. Also, we considered the sentences which have at least one word in the list of words with top ten highest TF-IDF scores, important.

Sentence position, a very different feature compare to TF-IDF, is a cue feature especially when the structure of the lecture is considered. For example: typically, professors welcome everyone in the audience which is usually the first sentence. This means, that the first sentence in the lecture is typically content irrelevant. The following sentences become more lecture-specific. At the end of lecture professors usually say goodby which is the last sentence of the talk and is also content irrelevant.

Sentence length is another feature of measuring the informativeness of a sentence. Very short sentences consisting one to three words used to be uninformative. Very long sentences sometime confuse the readers too. A length between five and twenty five words can be a good indicator of identifying informative sentences.

Another interesting feature is **the length of the word**. Words consisting only one or two characters tend to be articles, conjunctions and prepositions. The sentences with longer words are considered to be more informative. The sentences which have at least one word with length bigger than four are considered important.

Cue words consisting of unigrams, bigrams or trigrams are good indicators of important sentences. Such phrases as "The conclusion is" or "In summary" point out that the sentence concludes or summarizes the topic. An important information usually follows this phrases. The list of those key phrases is extracted from large data set using RSI metric described in chapter 4.3.2.1.

One of the most meaningful features in summarization is **the centroid feature** which is calculated for each cluster of sentences. In our case the segments of lecture are the clusters and the sentences are the elements of that cluster. The centroid of a cluster is constructed based on words having a $TF * IDF$ score higher than the predefined threshold. To find the distance between a sentence and cluster center we represent both as a vector of TF*IDF scores and calculate cosine similarity between them as shown in Equation 4.10.

4.4.2.1 Feature Representation

In order to use the features in summarization approaches we label them with different discrete values. Table 4.5 shows all features and their label-values which are assigned based on predefined thresholds. Cosine Similarity between a sentence and cluster center is computed based on the Equation 4.10

Table 4.5: The list of all summarization features and their values based on predefined thresholds

Feature	**Values**
Relative-Sentence-Position	Normalized, equally distributed ranges between [0..1]
Sentence-Length	Discrete numbers from 1 to N, where N is the number of words in sentence Too-Short(0), $Length \leq 3$ Middle-Length(1),$3 \leq Length \leq 50$ Too-Long(2), $Length < 50$
Word-Average-Length	Average number of characters in each word for given sentence Too-Short(0), $Length < 3$ Middle-Length(1), $3 \leq Length \leq 15$ Too-Long(2), $Length > 15$
TF*IDF-Avg	good(1), $TF * IDF \geq AVG_{TF*IDF}$ poor(0), $TF * IDF < AVG_{TF*IDF}$
TF*IDF-Max	good(1), $TF * IDF \geq MA\underline{X}_N(\{TF * IDF\})$ poor(0), $TF * IDF < MAX_N(\{TF * IDF\})$
Sentence-Similarity	*Too-Similar(0)*, when the sentence is identical with one of N successive sentences *Average-Similar(1)*, when there is at least one matching content word with any of N successive sentences *Too-Different(2)*, when there is no matching content words with any of N successive sentences
Centroid-Based	Far(0), $CosineSim(Sentence, Cluster_{Center}) < \theta$ Near(1), $CosineSim(Sentence, Cluster_{Center}) \geq \theta$
Cue Words	bigrams and trigrams extracted by RSI 4.3.2.1

5. Implementation

In this chapter we illustrate the implementation details of our system described in Chapter 4. For the implementation we used C and C++ programming language accompanied by a few *perl* scripts for system evaluation. We used additional libraries and tools such as *boost*-Library, *STL*, *libstemmer*-Library [Madd], *Part-of-Speech TreeTagger* [Schm], *libSVM* [ChLi11], *CRF++* [Chas] to support our implementation. In following paragraphs we describe the implementation details of each individual method.

5.1 Segmentation Component

For the segmentation we implemented three different algorithms, K-Means [VaUn07], HMM [aYou07] and Hierarchical Segmentation [WPZZ11]. All algorithms use segmentation features described in chapter 4.3.2. While K-Means and HMM use all three features described in 4.3.2, Hierarchical segmentation uses only content similarity. In the next chapters we describe the details of individual algorithms.

In our implementation the cluster centers in K-Means are initialized considering relative position of sentences. Cluster center are initialized as following:

Introduction $\rightarrow \frac{0.01*N}{2}$
Methods $\rightarrow \frac{0.2*N}{2} + 0.01 * N$
Questions $\rightarrow \frac{0.7*N}{2} + 0.21 * N$
Logistic Information $\rightarrow \frac{0.05*N}{2} + 0.91 * N$
Conclusion $\rightarrow \frac{0.05*N}{2} + 0.96 * N$

where N is the number of sentences in document.

It is also important to mention that in our implementation K-Means stops after 50 iterations or when the cluster centres do not change their positions any more.

In contrary to K-Means, HMM does not observe the probabilities of individual sentences but the group of sentences. We combine sentences in a group of three to four and estimate the probabilities for each group in each possible state. After probabilities for all sentence groups in all states are estimated the sentences are assigned to the state where they have the highest probability.

Table 5.1: An example section of SVM training file

$Label$	$S-Len$	TF_{Avg}	S_{Pos}	IDF_{Avg}	$TFIDF_{Avg}$	$TFIDF_{Max}$	
0.0	1:1	2:1	3:0	4:1	5:0	6:0	...
0.0	1:1	2:0	3:1	4:1	5:0	6:0	...
...							
1.0	1:1	2:1	3:13	4:1	5:0	6:0	...
1.0	1:1	2:1	3:14	4:1	5:0	6:0	...
1.0	1:0	2:1	3:15	4:1	5:0	6:0	...
...							

Hierarchical segmentation is a very different approach compare to K-Means and HMM. It allows to segment the documents into arbitrary number of segments using sliding window. The size of sliding window varies depending how fine granular we want to segment. In our implementation we halve the window size when we move to the next deeper level in hierarchy. For example if in level zero the window size is 50, in level one the window size is 25, in level two 12.

5.2 Summarization Component

Summarization module described in Chapter 4.4.1 offers two machine learning algorithms for summarization: SVM and CRF. Both algorithms are supervised and use the same feature set for training and testing. The system allows to configure desired algorithm before bootstrapping. To realize summarization based on SVM we used, *libSVM*, a open source library for Support Vector Machines. To use training and testing data in SVM we need to represent each sentence through a feature vector and scale it in a predefined scaling range before application. The training set before scaling is shown in table 5.1.

First column corresponds to binary label (*0:0* $\rightarrow$ *Non-Summary, 1:0* $\rightarrow$ *Summary*). The rest of the columns correspond to features. Each column represents one feature and its value subject to following convention: <feature-id>:<feature-value>. Testing data set has almost the same format as training data set only in the first column we use the same class either '1.0' or '0.0'. Since we do not know the exact data labels.

For our experiments we trained four different models. One model for whole lecture without considering segmentation and three models, one for each three rhetorical segment(Introduction, Content and Conclusion).

In the testing phase we call SVM's predict function on scaled lecture data to predict the labels of unseen lectures. The output of this step is the probability of each sentence being classified to "Summary" class. The sentences with the highest probability are selected based on summarization ratio.

To implement summarization based on CRF we used CRF++ open source library. Similar to SVM, during training CRF model is learned based on feature vectors of training data. And during testing we label unseen lectures based on model learned during training phase. Table 5.2 shows some features and example values. In training and test data file the first column corresponds to sentence id, the last

Table 5.2: An example section of CRF training file

$Sent_{Id}$	$S-Len$	TF_{Avg}	S_{Pos}	...	$Label$
0	$MIDDLE\_LEN$	1	0_0.04	...	0.0
8	$MIDDLE\_LEN$	1	0_0.04	...	0.0
9	$MIDDLE\_LEN$	1	0_0.04	...	0.0
...					
2381	TOO_SHORT	1	0.96_1	...	1.0
2383	$MIDDLE\_LEN$	1	0.96_1	...	1.0
2384	$MIDDLE\_LEN$	1	0.96_1	...	1.0

column is the label and the other columns represent the feature values. The feature set used both for SVM and CRF are the same only in different formats. The only difference is that CRF also considers the sequences of sentences which are predefined in a template file.

6. Evaluation

Techniques of automatically evaluating summaries are classified into two categories: intrinsic and extrinsic. Intrinsic techniques consider the informativeness, coherence and redundancy of the summary. In contrast to intrinsic, extrinsic techniques measure query and topic relevance of the summary.

In this work intrinsic approaches are applied only. One of those approaches is ROUGE (Recall Oriented Understanding of Gisting Evaluation) which relies on n-gram matches between automatic and manual summaries. Although ROUGE is the most popular metric among automatic evaluation metrics in text summarization, it is still imperfect since only n-gram word matches could not be a perfect measure of summarization quality. Further intrinsic approaches which we used for segmentation are Precision, Recall, F-Measure and P_k-Measure.

In order to evaluate the performance of summarizer and segmenter we need ground truth summaries and segments. However, manual evaluation of summaries and segments varies strongly from human to human and is labour intensive. Although there are some correlations between manual evaluations the evaluation is still very subjective, therefore inter-annotator agreement is low. One idea to solve this problem could be gold standard summaries. However, to produce gold standard summaries we need multiple annotators summarizing the same lecture which is time consuming and leads to high costs.

6.1 Corpus

In experiments we used two corpora. The first corpus comes from lecture domain and is used for evaluating both segmentation and summarization. In this dataset most lectures are technical, around 60+ lectures held at the Karlsruhe Institute of Technology (KIT). The second corpus comes from scientific paper domain, encompasses 3000+ papers and is used only for evaluating the segmentation.

6.2 Human Annotation

Four different human annotators manually segmented and summarized transcribed lectures. Manual segmentation considered three and five segments and summarization encompasses long and short summaries. In manual segmentation the annotators split the segments from each other using line break and in summarization for each lecture the annotators create short and long summaries. Short summaries capture 20 percent and the long summaries only 80 percent sentences of the original text. The summaries are created based on predefined instructions described in section 6.2. These instructions are focused on sentence extraction but also require to eliminate redundant phrases, words or disfluency. Furthermore, for the evaluation of word selection based summarization we manually create short and long summaries by extracting only words and combining them together. Table 6.1 shows the annotated corpus and its features. The column **Lecture** of table 6.1 shows the field of science that the lectures come from. Those are: computer science, history, math and law. Column **Nr. of Sentences** referred to number of sentences both in English and German transcripts. In addition to summarization, we also annotated the lectures

Table 6.1: The list of annotated lectures used for 10-Fold evaluation

Lecture	**Duration**	**Nr. of Sentences**	**Nr. of Words in English**	**Nr. of Words in German**
Comp. Sc.	5.31 min	53	797	785
Comp. Sc.	83.35 min	810	11,963	11,926
Comp. Sc.	41.63 min	337	4,109	3,759
Comp. Sc.	67.01 min	471	10,304	9,150
Comp. Sc.	18.47 min	107	2,317	2,059
Comp. Sc.	6.23 min	53	659	597
Law	61.83 min	1,071	13,176	12,637
History	59.48 min	481	7,985	7,435
Math	5.33 min	49	502	487
Math	80.19 min	385	7,077	6,631

into different segmentation units. The segmentation units are described in section 4.3.1.

Reference Summary based on Sentence Extraction

Reference summary is well-organized and extracted from original text by removing redundant information. First, the informative sentences are extracted. Those are the sentences which convey the main ideas and key points such as definitions, advantages, disadvantages mentioned in the talk. Signal words or phrases such as 'and the important thing here is ...' or 'first I would like.... second .... ' help to find the key points of the topic. Second, if the extracted sentences contain disfluency such as false starts, hesitations, repetitions, filled pauses interruptions or wrong sentence boundaries, they are corrected with minimum changes. In case the sentences are too long, only main or sub-ordinary clause are extracted. During summarization no specialized knowledge is used to draw conclusions or make inferences that are not obvious in the lectures. Bellow we can see an example annotation of a lecture.

...
@but it is more about seeing how the rule+ application+ works with it
@the first rule+ <that has to be used> is the non-recursive+ application+ of updates+
@that is replica U+ is applied+ to a function+ that starts with G on the argument and there we have to use the U+ recursively+
@and from now on it continues+ non-recursively+<, how it is written here>
this is easy in this case
@I did not write down which rules+ apply+ here
...

In this example the important sentences are marked with '@' symbol and the content relevant words with '+' symbol after each salient word. The unnecessary words and phrases are marked in '<' and '>' symbols.

Reference Summary based on Word Extraction

Similar to reference summary based on sentence extraction as described in section 6.2 the summary based on word extraction is well organized and extracted from original text. The only difference is that here instead of selecting individual sentences we combine three or four sentences as one sentence. To combine the sentences we extract the words and put them together thereby creating a new sentence. Bellow we can see an example manual annotation based on word extraction.

...
it is more about seeing how the rule application works with it first rule that has to be used is the non-recursive application updates
replica U is applied to a function that starts with G on the argument and there we have to use the U recursively from now on it continues non-recursively
and from now on it continues how it is written here is easy in this case
there are again at least two that apply here but we can see directly that here it is zero
...

As we can see from the summary, two to three sentences are combined together to form a new sentence. Newly formed sentence is in many cases grammatically incorrect. From here we can infer that the combination of many sentences does not always produce grammatically correct sentence.

6.3 Evaluation Metric

In this chapter we introduce the evaluation metric used for evaluating the performance of summarization and segmentation methods. ROUGE [Lin04] is the most popular metric for evaluating the quality of summaries. It compares human annotated summaries to automatic generated ones based on N-gram matches. ROUGE score has different types of measure: ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S. ROUGE-N considers N-gram matches, ROUGE-L longest common sequence, ROUGE-W stands for weighted longest common sequence, ROUGE-S Skipped-bigram co-occurrence statistics and ROUGE-SU* which is an extension of ROUGE-SU. ROUGE-N is computed based on Equation:

$$ROUGE - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)} \tag{6.1}$$

We first calculated ROUGE -recall, -precision and -F-measure (harmonic mean) but used only F-Measure of ROUGE score to evaluate and compare the performance of

summarization systems. Since F-Measure is the optimal combination of precision and recall.

In contrary to summarization, segmentation accuracy is not measured with ROUGE score. In order to estimate segmentation accuracy we used metrics such as recall, precision, F-Measure and P_k [BeBL99]. P_k is a probabilistic error metric based on given window size and nearness to segment boundaries. It shows the probability that a randomly chosen pair of words k is inconsistently classified in segments. In our experiments the size of k is set to half of average segment length. The values of P_k lies between [0,1] (The smaller P_k value the better).

6.4 Evaluation Experiments

We accomplished experimental evaluation both for segmentation and summarization based on various approaches. In segmentation, going out from the fact that lecture data is insufficient we additionally used large number of scientific papers to train and test segmentation module. Section names are used for automatic segmentation of scientific paper. In contrary to segmentation, summarization experiments are accomplished using only lecture data. The reason for that is the fact that for scientific papers we do not own reference summaries (ground truth) and automatic extracted summaries are very abstract. In the following sections we discuss experiments in detail.

6.4.1 Experimental Setup

In all segmentation experiments we partition documents in five segments using three methods: K-Means, HMM and Hierarchical Segmentation. The similarity measure used both in K-Means and HMM relies on tri-grams and two-grams extracted by RSI(4.3.2.1) metric.

Since some of our approaches use hyper-parameters such as α, β, λ_1 and λ_2, we tried different combinations of those parameters and chose those which lead to best performance. In order to find the best combination of α and β for similarity measure we applied cross-validation and tried different possible combinatorial values. The experiments showed that α equal to 0.2, β equal to 0.3 and respectively $1 - \alpha - \beta$ equal to 0.5 are the best possible combinations. We applied the same technique to find the best combination of λ_1 and λ_2 for computing RSI. According to our experiments the best results are achieved by using λ_1 equal to 0.9 and λ_2 equal to 0.1.

The hierarchy depth in Hierarchical clustering is set to 2. The window length for the first hierarchy level is set to 50 and for the second is set to 25. Error window in all segmentation evaluation methods is set to 6.

Our SVM based summarizer uses epsilon Support Vector Regression (SVR) with Radial Basis Function (RBF) kernel. The hyper-parameter, C (cost function) is set to 1 and γ to 1/2.

The details about the data sets are described in chapters 6.4.1.1 and 6.4.1.2.

6.4.1.1 Experiments based on Lectures

Our lecture dataset consists of 70 lectures: 63 lectures are used for training and 7 for testing. Before evaluation, we first manually annotate individual segments of lectures and train the system based on 63 lectures. The tests are run afterwards on trained model and the resulted segments are evaluated using 7 manually annotated lectures.

6.4.1.2 Experiments based on Scientific Papers

First, we download 3.500+ scientific papers from ACL's conference proceeding archive [1], convert them into text and clean up all titles, equations, images, labels, references to simulate the running text stream as lectures. These papers are of very similar domain and relatively similar structure in presentation. For example most of them have abstract, introduction, conclusion and evaluation. Second, we train our segmentation module based on 3400+ cleaned up papers and segment unseen papers based on learned model. Third, we run 30-Fold cross-validation evaluation to estimate the performance of our system. As we mentioned in the beginning of this chapter scientific papers are used for segmentation only.

6.4.2 Segmentation Results

Segmentation evaluation results according precision, recall, F-measure using error window with size 6 and P_k metric on scientific papers are listed in Table 6.2 and lectures in Table 6.3. These results show that the segmentation based on HMM has the highest evaluation score. We also observe that RSI feature has significantly improved the accuracies of segmentation on precision, recall and F-Measure scores. Generally, hierarchical segmentation achieves higher score than K-Means and K-Means+RSI but lower score than HMM. Additionally, we also observe that the usage of RSI improves HMM precision, recall and F-Measure scores more than K-Means score. However, the improvements of P_k score are comparably small for all methods. For scientific papers RSI does not improve HMM P_k score at all and we observe only slight improvements for other scores. Yet, hierarchical segmentation is computationally much cheaper than HMM and is more likely to be implemented as an online segmentation system compared to HMM or the Hierarchical HMM (HHMM).

It was also interesting to observe that in general random segmentation has comparably lower score than HMM, K-Means and Hierarchical Segmentation.

In general segmentation evaluation scores for scientific papers outperform evaluation scores for lectures. This is because the ground truth of segmentation evaluation is objective and the data set is larger compare to lectures.

6.4.3 Summarization Results

We run our experiments on SVM, CRF, Word Selection based summarizers (in Chapter 4.4.1.1), Lemur MMR (in Chapter 2.1.1.2), Lemur Basic (in Chapter 2.1.1.2), OTS (in Chapter 2.1.1.2) and random sentence extraction summarizer. During experiments we considered both structure-dependent and structure-independent, long

[1] http://aclweb.org/anthology-new/

Table 6.2: Segmentation evaluation results for scientific papers both with and without Rhetorical Structure Index (RSI) based on HMM, K-Means and Hierarchical segmentation.

Method	K-Means	K-Means +RSI	HMM	HMM +RSI	Hierar. Segment.	Random
Precision	36.67	37.61	69.72	**72.44**	38.69	30.41
Recall	40.69	41.80	36.74	**44.67**	42.72	34.56
F-Measure	38.24	39.27	48.12	**51.61**	40.60	32.41
P_k	0.09	0.07	0.03	**0.01**	0.06	0.40

Table 6.3: Segmentation evaluation results for lectures both with and without Rhetorical Structure Index (RSI) based on HMM, K-Means and Hierarchical segmentation.

Method	K-Means	K-Means +RSI	HMM	HMM +RSI	Hierar. Segment.	Random
Precision	33.33	33.34	58.14	**60.34**	37.11	31.28
Recall	40.00	40.21	39.86	**40.15**	43.12	38.05
F-Measure	36.36	36.45	47.29	**48.21**	39.88	34.33
P_k	0.15	0.11	0.05	**0.03**	0.09	0.43

(summarization ratio: 80%) and short (summarization ratio: 20%) summaries. Each summarization approach is applied twice on a lecture. First, summarization approach is applied on individual segments of lecture and the summarized segments are put together building the entire summary. Second, the summarization approach is applied on entire lecture without considering segmentation. Tables 6.4 and 6.5 show the evaluation results of each summarization method with and without segmentation for the long summaries (summarization ratio: 80%). Tables 6.6 and 6.7 show the results for the same system parameters but for the short summaries (summarization ratio: 20%) respectively with and without segmentation. Summarization evaluation, in all experiments, considers different variants of ROUGE score.

Table 6.4: Comparing ROUGE average F-Measure scores for five various summarization approaches **without applying segmentation.** Summarization ratio: 80%

Avg_F: 80%	OTS	Basic	MMR	Hori's-Sum	SVM	CRF	Random
ROUGE-1	86.23	**89.84**	78.90	87.48	85.49	87.76	78.10
ROUGE-2	80.12	81.88	69.16	81.51	79.22	**82.49**	67.65
ROUGE-3	77.99	80.24	64.32	74.73	76.58	**80.55**	61.11
ROUGE-4	76.22	72.82	61.44	68.73	74.72	**78.97**	56.24
ROUGE-L	86.23	79.84	78.76	87.48	85.23	**87.52**	84.28
ROUGE-W-1.2	32.59	**34.26**	26.90	**34.26**	29.94	32.38	24.78
ROUGE-S*	75.01	76.17	46.37	77.85	73.85	**78.03**	41.12
ROUGE-SU*	75.04	76.19	46.46	77.87	73.88	**78.05**	45.06

In general, CRF appears to have better summarization scores both with and without segmentation compared to SVM and other methods. The best summarization

Table 6.5: Comparing ROUGE average F-Measure scores for five various summarization approaches **with segmentation.** Summarization ratio: 80%

Avg_F: 80%	OTS	Basic	MMR	Hori's-Sum	SVM	CRF	Random
ROUGE-1	87.35	90.34	90.63	88.79	85.75	**91.10**	83.15
ROUGE-2	81.55	86.13	81.22	79.93	79.48	**88.06**	79.56
ROUGE-3	79.06	84.60	70.96	71.87	76.58	**86.76**	71.45
ROUGE-4	77.31	83.30	72.58	65.31	74.72	**85.58**	62.56
ROUGE-L	87.35	90.34	90.63	88.67	85.49	**91.10**	85.50
ROUGE-W-1.2	32.80	34.00	30.85	33.74	29.42	**34.78**	27.11
ROUGE-S*	77.24	81.99	72.64	80.09	74.37	**83.43**	74.75
ROUGE-SU*	77.27	82.01	72.69	80.11	74.40	**83.45**	71.36

Table 6.6: Comparing ROUGE average F-Measure scores for five various summarization approaches **without applying segmentation.** Summarization ratio: 20%

Avg_F: 20%	OTS	Basic	MMR	Hori's-Sum	SVM	CRF	Random
ROUGE-1	62.11	**65.10**	42.63	35.29	58.19	58.87	42.50
ROUGE-2	50.40	50.51	13.20	09.08	33.66	**55.62**	11.69
ROUGE-3	46.63	46.28	07.63	01.21	27.18	**50.65**	08.20
ROUGE-4	44.84	44.41	06.05	00.00	24.17	**49.80**	06.98
ROUGE-L	61.32	**64.09**	39.84	35.29	56.65	56.53	33.29
ROUGE-W-1.2	19.64	**27.43**	15.43	13.20	19.93	20.31	12.01
ROUGE-S*	**43.32**	41.67	16.89	11.82	35.23	35.52	14.11
ROUGE-SU*	**42.05**	41.83	17.10	11.96	35.40	35.70	16.54

is achieved by CRF with Segmentation. This is connected with the fact that unlike SVM, CRF not only considers the features of each individual sentence but also their order therefore fits better to this task. Specifically, we observe that for 80% summarization ratio without segmentation Lemur-basic for unigrams and Hori's method for weighted ROUGE score show better performance than CRF. We also observe that OTS outperforms CRF for short summaries without segmentation in ROUGE skipped bigram and skipped bigram extension and that Lemur Basic outperforms CRF in ROUGE-1, ROUGE-L and ROUGE-W-1.2. After evaluating the results for short summaries with segmentation we observe that Lemur-Basic and OTS show the best performance in many ROUGE scores outrunning CRF.

Also we compared different summarizaton methods and their performance to each other. For example, when we look at summarization results of Chiori's method, we observe average performance for long summaries and low performance for short summaries. Even random sentence selection based summarizer outperforms it. This is connected with the fact that during compression we are obligated to select a limited number of words and are confronted with the lack of selection. Chiori's method also strongly depends on the dataset used in the training. Since linguistic score entirely depends on the word combinations learned from the training set as described in Chapter 4.4.1.1.

Additionally, we observed that even such simple methods as Lemur-Basic and OTS show relative gut performance compared to Chiori's method, Lemur-MMR, SVM- and CRF-based summarization approaches. In general for long summaries Lemur-

Table 6.7: Comparing ROUGE average F-Measure scores for five various summarization approaches **with segmentation.** Summarization ratio: 20%

Avg_F: 20%	OTS	Basic	MMR	Hori's-Sum	SVM	CRF	Random
ROUGE-1	66.52	67.63	62.82	35.82	63.24	**67.95**	61.83
ROUGE-2	54.79	**54.93**	47.11	07.87	41.65	52.72	42.20
ROUGE-3	50.56	**51.54**	43.69	01.88	37.27	47.97	36.99
ROUGE-4	48.99	**49.76**	42.25	00.00	36.18	46.02	37.11
ROUGE-L	54.75	66.99	61.23	34.7	61.60	**67.25**	56.58
ROUGE-W-1.2	**30.29**	26.33	29.17	12.40	27.13	28.35	14.65
ROUGE-S*	41.89	43.91	36.16	12.00	40.27	**44.24**	38.25
ROUGE-SU*	43.52	44.06	36.37	12.18	40.46	**44.41**	37.78

Basic is significantly better than OTS. The lower performance of Lemur-MMR is connected with its query nature. Lemur implementation of MMR uses the first sentence of lecture as a query which used to be a welcoming sentence and doesn't have anything to do with the content of lecture. The performance of SVM also lies behind CRF, OTS, Lemur-Basic. This means that a simple sentence extraction algorithm based on word frequencies outperforms a sentence extraction based on classifier. One reason for this is the annotated lecture summaries. Since we have sparse annotated lecture summaries our SVM model is not optimally trained.

In general all sentence extraction based approaches show better performance than random sentence extraction. Yet, in some cases random extraction outperforms SVM for short summaries.

Also, comparing the results in Table 6.4 with the results in 6.6 or the results in Table 6.5 with the results in 6.7 we observe that short summaries in general have lower score than long ones. This is connected with the fact that the matches between reference and hypothesis in long summaries are more probable, therefore an "easier" task compared to short summaries.

Word Extraction based Summarizer

We implemented word extraction based summarizer adapting Chiori's approach [HFMY+02]. The summarizer was trained and evaluated based on data sets of different genre and measures of information importance. Table 6.8 shows the list of data sources used for training and their size. Tables 6.12 and 6.13 show the summarization evaluation results after applying three different training data sets. In

Table 6.8: The list of all training data sets used by word extraction based summarizer.

Data	Size
europarl	232 MB
btec	3.5 MB
ted	4.3 MB

order to identify qualitatively best training data from those listed in Table 6.8, we calculated the perplexity of each summary for language models trained on different

training sets. Table 6.9 shows the list of perplexities both for automatically generated and manually created summaries for different summary length and type. The perplexities show that the summary has the lowest value, trained on TED corpus. This means that TED is the qualitatively best corpus since it has the lowest perplexity, therefore also the lowest branching factor. Specifically, for TED corpus we applied topic segmentation and used the individual topics as documents. The results are illustrated in Table 6.10. The results show that by using topics as documents we reach better performance. This is because the topics are better indicator of documents than sentences. Sentences are too short to be considered as documents.

Table 6.9: Perplexities for summaries applied on various training sets

ppl(summary|lm)	btec-lm	ted-lm	europarl-lm
Word-extr-btec-80%-sum	0762.77	**0222.13**	0489.98
Word-extr-btec-20%-sum	1506.93	**0657.98**	3091.69
Word-extr-ted-80%-sum	0779.52	**0207.20**	0503.95
Word-extr-ted-20%-sum	2050.81	**0782.45**	3500.27
Word-extr-europarl-80%-sum	0723.08	**0245.66**	0443.73
Word-extr-europarl-20%-sum	1736.14	**0816.84**	2437.66
Human-translation	0862.76	**0226.92**	0652.34
Human-sum-sent.extr.80%-sum	0762.33	**0252.77**	0635.44
Human-sum-sent.extr.20%-sum	1066.32	**0504.45**	1639.47
Human-sum-sent.extr.80%-sum	0747.86	**0342.36**	0631.97
Human-sum-sent.extr.20%-sum	4171.37	**2654.88**	8479.59

Table 6.10: Comparing Average ROUGE F-Measure score (summarization ratio: 80%) for Word extraction based summarizer using TED corpus as training set considering in one case sentences and in other case topics as documents.

Avg_F: 80%	Hori-TED(4.3 MB) – Each sentence as a Document	Hori-TED(4.3 MB) – Each topic as a Document
ROUGE-1	86.17	**87.90**
ROUGE-2	72.59	**75.70**
ROUGE-3	61.99	**66.32**
ROUGE-4	52.72	**58.87**
ROUGE-L	86.17	**87.60**
ROUGE-W-1.2	30.89	**31.33**
ROUGE-S*	74.49	**77.25**
ROUGE-SU*	74.52	**77.28**

Although TED corpus has the lowest perplexity it does not appear to show the best results for summarization as shown in tables 6.12 and 6.13. This is to explain with the fact that for example europarl corpus is comparably larger than TED therefore it shows better results. However, it is interesting to observe that using btec corpus, the smallest corpus, leads to the best results for short summaries compared to TED and europarl. And according to our experiments btec appears to outperform TED corpus in all cases. This leads us to the assumption that Chiori's method strongly depends on the domain and genre of data set. For example, TED lectures are informal and less structured.

Table 6.11: Comparing Average ROUGE F-Measure score (summarization ratio: 20%) for Word extraction based summarizer using TED corpus as training set considering in one case sentences and in other case topics as documents.

Avg_F: 20%	Hori-ted(4.3 MB) – Each sentence as a Document	Hori-ted(4.3 MB) – Each topic as a Document
ROUGE-1	45.83	**47.24**
ROUGE-2	13.65	**19.76**
ROUGE-3	02.74	**08.33**
ROUGE-4	00.39	**03.59**
ROUGE-L	45.05	**46.85**
ROUGE-W-1.2	18.63	**20.28**
ROUGE-S*	20.33	**22.09**
ROUGE-SU*	20.51	**22.28**

Table 6.12: Comparing ROUGE average F-Measure scores for word extraction based summarizer using three different training corpora: btec, ted and europarl. Summarization ratio: 80%

Avg_F: 80%	Hori-btec(3.5 MB)	Hori-ted(4.3 MB)	Hori-eparl(232.1 MB)
ROUGE-1	85.89	86.17	**86.82**
ROUGE-2	73.90	72.59	**74.42**
ROUGE-3	63.37	61.99	**64.08**
ROUGE-4	54.46	52.72	**55.20**
ROUGE-L	85.76	86.17	**86.37**
ROUGE-W-1.2	30.74	30.89	**32.05**
ROUGE-S*	74.07	74.49	**75.66**
ROUGE-SU*	74.10	74.52	**75.69**

The training data set listed in table 6.8 was used for learning language model parameters. To build the language model we used SRI language model (SRILM) [Stol02] toolkit together with Katz's n-gram back-off model. Katz's backoff model is especially useful when the history is small. It allows to smooth the language model values in case the history is unavailable.

Moreover, we applied different methods for computing the significance score of a word in summary. In the first experiment we used frequency of content words as significance score. Second experiment considered inverse term frequency, in the third experiment we applied term frequency inverse term as a significance score(in Chapter 2.1.1.1) and in the last experiment we used entropy as a significance score. The experimental results confirmed our assumption about TF*IDF being the best measure of significance. In all these experiments we assumed that each sentence is a document.

In general segmentation and sentence extraction based on different features shows comparable good results for all approaches. We shows that segmentation improved summarization quality and that RSI significantly improved segmentation quality and that TF*IDF is not perfect but one of the best metrics to measure the quality. Also, numerous experiments for Word Extraction method showed that it tends to result in lower performance than sentence extraction based methods.

Table 6.13: Comparing ROUGE average F-Measure scores for word extraction based summarizer using three different training corpora: btec, ted and europarl. Summarization ratio: 20%

Avg_F: 20%	Hori-btec(3.5 MB)	Hori-ted(4.3 MB)	Hori-eparl(232.1 MB)
ROUGE-1	46.99	45.83	**47.56**
ROUGE-2	**16.34**	13.65	15.66
ROUGE-3	**05.47**	02.74	03.54
ROUGE-4	**02.75**	00.39	00.04
ROUGE-L	46.51	45.05	**46.78**
ROUGE-W-1.2	**19.84**	18.63	19.31
ROUGE-S*	20.37	20.33	**21.32**
ROUGE-SU*	20.55	20.51	**21.50**

7. Conclusion

Significant advancements in distance learning and growing number of online lectures increases the demand for structuring and summarizing lecture transcriptions so that students can retrieve required information easily. The problem of how to structure and summarize lecture transcriptions has many challenges that need to be addressed. Lecture transcriptions often contain repeated information and inaccurate structure. Transcriptions generated by an automatic speech recognizer (ASR), contain speech disfluency, errors caused by the speech recognizer and sentence boundaries that are often missing. However, although lecture transcriptions are not as structured as news for example, they do usually contain a certain rhetorical structure. For example, lectures usually start with "Introduction" followed by "Background Information", "Methods" and end with a "Conclusion". There are already existing approaches to spoken lecture summarization and segmentation; however, being young research fields, both segmentation and summarization exhibit large room for extensions and improvements.

In this thesis we investigated automatic methods for spoken lecture segmentation and structure-dependent summarization to enhance the usability of online lectures. We proposes a novel Rhetorical Structure Index (RSI) to measure the "structural importance" of a word or phrase. We applied the RSI together with other structural and lexical features in three different segmentation methods including K-Means, Hierarchical Segmentation and HMM and showed that with the discovered structure, the summarization system can be significantly improved. Summarization methods include SVM and CRF, two different supervised machine learning approaches. Both methods showed significant improvements when they are used together with segmentation.

7.1 Contribution of the Thesis

One of the most important contributions of this work is the novel metric, Rhetorical Structure Index (RSI), which allows us to measure the structural importance of words or phrases. The RSI is based on weighted position variance and occurrence frequency of words or phrases. Using the RSI, we extract words and phrases which

are indicators of different structural units. Extracted words or phrases are used later as additional features in summarization and segmentation systems. The RSI shows significant improvements in summarization and three segmentation methods: K-Means, HMM and Hierarchical Segmentation.

Another contribution of the thesis is the application of hierarchical segmentation. We applied sliding window at each position of the document and computed content similarities between two windows. This way we could identify the positions, "peaks", where the contents are very different. Peak positions are potential candidate positions of topic changes and therefore a new segment indicators. We hierarchically apply smaller sliding windows in individual segments to find fine, granular sub-segments. This approach showed significant improvements compared to K-Means.

A further contribution of this thesis is the application of segmentation together with Conditional Random Fields (CRF) for spoken lecture summarization. CRF allow us to classify the sentences in "summary" and "non-summary" classes by using a large number of summarization features. In contrary to many other classification methods, CRF takes into consideration the sequences of sentences and is able to handle large number of features which contributes CRF to outperform existing summarization approaches. Segmentation together with CRF showed the best performance compared to any other method.

7.2 Discussion and Future Work

As mentioned in the beginning of this chapter we developed a system for spoken lectures segmentation and summarization which produces concise summaries and partitions whole lecture into logical segments. For segmentation we developed three different methods: K-Means, Hierarchical Segmentation and HMM. HMM appears to show the best performance compared to other methods. Compared to K-Means and HMM, Hierarchical Segmentation allows us to have an arbitrary number of segments; therefore, it is more flexible. Segmentation showed significant improvement contribution also in summarization. For summarization we evaluated both existing and new developed summarization software. Summarization methods mostly cover extraction approaches. But we also implemented one already existing word selection based summarization approach proposed by Chiori [HFMY+02] in 2003. Overall evaluation of all systems showed that word extraction based summarization performance is inferior to sentence extraction which correlates with human opinion as shown in Section 4.4.1. Between sentence extraction approaches OTS and Lemur-Basic show comparable good performance. They even outperform SVM based summarizer. Due to query restrictions Lemur-MMR showed worse results compared to other methods. The Lemur toolkit uses the first sentence in a document as a query, in case it is not specified. Overall the best performance in summarization showed CRF based extractive summarizer using segmentation.

In our work one of the possible future directions could be the extension of proposed HMM model. The new model proposes to integrate summarization and segmentation models in one HMM model. We can for example have HMM states which only emit summary sentences. In this case each HMM state can be represented as an integrated state of segmentation and summarization. For example, the segmen-

tation label *Introduction* can be connected with two HMM states. One state can be called *Introduction+Summary* and the other one *Introduction+Non-Summary*. These means that at the end we will have twice as many HMM states as now. But our system will be much simpler and we can combine and examine both segmentation and summarization parameters together.

Another future direction can be the integration of prosodic characteristics which can help to detect important segments of lectures. We can use prosodic variations such as pitch, intensity, duration as additional features in our statistical model. These features can be extracted from audio lectures.

Literatur

[aYou07] M. G. andSteve Young. The Application of Hidden Markov Models in Speech Recognition. 2007.

[BBBC+] A. Balagopalan, L. L. Balasubramanian, V. Balasubramanian, N. Chandrasekharan und A. Damodar. Automatic Keyphrase Extraction and Segmentation of Video Lectures.

[BeBL99] D. Beeferman, A. Berger und J. Lafferty. Statistical Models for Text Segmentation. *Mach. Learn.* 34(1-3), Februar 1999, S. 177–210.

[CaGo98] J. G. Carbonell und J. Goldstein. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *SIGIR*, 1998, S. 335–336.

[Chas] T. Chasen. CRF++: Yet Another CRF toolkit. http://crfpp.googlecode.com/svn/trunk/doc/index.html.

[ChLi11] C.-C. Chang und C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST* 2(3), 2011, S. 27.

[CHYL11] Y.-N. Chen, Y. Huang, C. feng Yeh und L.-S. Lee. Spoken Lecture Summarization by Random Walk over a Graph Constructed with Automatically Extracted Key Terms. In *INTERSPEECH*, 2011, S. 933–936.

[Coll] T. B. Colleges. Top 25 Colleges and Universities Offering Free Online Classes and Lectures. http://www.thebestcolleges.org/free-online-classes-and-course-lectures/.

[ErRa11] G. Erkan und D. R. Radev. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *CoRR* Band abs/1109.2128, 2011.

[HFMY+02] C. Hori, S. Furui, R. Malkin, H. Yu und A. Waibel. Automatic speech summarization applied to English broadcast news speech. In *ICASSP*, 2002, S. 9–12.

[Kozi96] H. Kozima. Text Segmentation Based on Similarity between Words. *CoRR* Band cmp-lg/9601005, 1996.

[LaMP01] J. D. Lafferty, A. McCallum und F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. 2001, S. 282–289.

[LiEf00] R. Lienhart und W. Effelsberg. Automatic Text Segmentation and Text Recognition for Video Indexing. *Multimedia Syst.* 8(1), 2000, S. 69–81.

[LiLi09] F. Liu und Y. Liu. From Extractive to Abstractive Meeting Summaries: Can It Be Done by Sentence Compression? In *ACL/AFNLP (Short Papers)*, 2009, S. 261–264.

[LiLi10] F. Liu und Y. Liu. Using spoken utterance compression for meeting summarization: A pilot study. In *SLT*, 2010, S. 37–42.

[Lin04] C.-Y. Lin. A Package for Automatic Evaluation of Summaries. *In Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL, Barcelona, Spain*, 2004, S. 1–10.

[LiXL10] Y. Liu, S. Xie und F. Liu. Using n-best recognition output for extractive summarization and keyword extraction in meeting speech. In *ICASSP*, 2010, S. 5310–5313.

[Madd] B. Madden. TreeTagger - a language independent part-of-speech tagger. http://snowball.tartarus.org/.

[MaRS08] C. D. Manning, P. Raghavan und H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA. 2008.

[oMUn] U. of Massachusetts und C. M. University. The Lemur Project. http://www.lemurproject.org/.

[21] N. I. of Standards und T. (NIST). Document Understanding Conference. http://duc.nist.gov/.

[22] N. I. of Standards und T. (NIST). Text Analysis Conference. http://www.nist.gov/tac/.

[PoCr97] J. M. Ponte und W. B. Croft. Text Segmentation by Topic. In *ECDL*, 1997, S. 113–125.

[Rote] N. Rotem. Open Text Summarizer. http://libots.sourceforge.net.

[Schm] H. Schmid. Snowball - Projects. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

[ShMi08] M. M. Shafiei und E. E. Milios. A Statistical Model for Topic Segmentation and Clustering. In *Canadian Conference on AI*, 2008, S. 283–295.

[SILR10] M. Scaiano, D. Inkpen, R. Laganière und A. Reinhartz. Automatic Text Segmentation for Movie Subtitles. In *Canadian Conference on AI*, 2010, S. 295–298.

[Stol02] A. Stolcke. SRILM - an extensible language modeling toolkit. In *INTERSPEECH*, 2002.

[Vapn95] V. N. Vapnik. The nature of statistical learning theory. 1995.

[VaUn07] S. Vassilvitskii und S. University. *K-means: Algorithms, Analyses, Experiments*. Stanford University. 2007.

[WaLX10] X. Wan, H. Li und J. Xiao. Cross-Language Document Summarization Based on Machine Translation Quality Prediction. In *ACL*, 2010, S. 917–926.

[Wan11] X. Wan. Using Bilingual Information for Cross-Language Document Summarization. In *ACL*, 2011, S. 1546–1555.

[WPZZ11] P. Wu, H.-K. Peng, J. Zhu und Y. Zhang. SensCare: Semi-Automatic Activity Summarization System for Elderly Care. In *the Proceedings of The 3rd International Conference on Mobile Computing, Applications, and Services (MobiCASE 2011)*, Los Angeles, CA, October 24-27 2011.

[ZeWa00] K. Zechner und A. Waibel. DIASUMM: Flexible Summarization of Spontaneous Dialogues in Unrestricted Domains. In *COLING*, 2000, S. 968–974.

[ZhCF07] J. J. Zhang, R. H. Y. Chan und P. Fung. Improving lecture speech summarization using rhetorical information. In *ASRU*, 2007, S. 195–200.

[ZhCF09] J. J. Zhang, R. H. Y. Chan und P. Fung. Extractive speech summarization by active learning. In *ASRU*, 2009, S. 392–397.

[ZhCF10] J. J. Zhang, R. H. Y. Chan und P. Fung. Extractive Speech Summarization Using Shallow Rhetorical Structure Modeling. *IEEE Transactions on Audio, Speech & Language Processing* 18(6), 2010, S. 1147–1157.

[ZhFu12] J. J. Zhang und P. Fung. Active learning with semi-automatic annotation for extractive speech summarization. *TSLP* 8(4), 2012, S. 6.

[ZhHF08] J. J. Zhang, S. Huang und P. Fung. RSHMM++ for extractive lecture speech summarization. In *SLT*, 2008, S. 161–164.

A. Appendix A

Example Segmentation and Summarization

An example lecture from machine learning class segmented using hierarchical segmentation. The segments are represented in separate boxes.

1. *yes .*
2. *today we want to get busy with the second lecture of machine learning , and we want to get busy here once again in somewhat more depth with a further class of learning algorithms , which all are collectively called neural networks , and we want to get familiar with one of the most popular approaches , the so - called backpropagation algorithm .*
3. *you can certainly also learn neuronal networks as a classifier for many classification tasks which we met already , which would belong in the group of non - parametric , non - linear classifiers , and thus is also offering a variation which represents non - parametric , non - linear classifiers , which thereby are of course especially interesting for the case of more complex , complicated classification tasks like those we have seen so far .*
4. *well now , to introduce this , first of all , I want to start once again with the perceptron learning algorithm .*
5.*we rushed through this last time in a big hurry , so that I think that it would be certainly useful once again , if we would re - sketch it , and then would deal with the so - called multilayer perceptron or the neural networks by building up on this .*
6.*now , the complete reason for , and all of the euphoria about neural networks , this was having its high time about ten years ago , I would say , ten , fifteen years ago .*

7. *at that time people got excited of perhaps being able to develop algorithms which would rather correspond to the computing strategies of the human brain than what is known from the Von - Neumann machines .*
8. *now , the Von - Neumann machine , which we all love as the PC , is a totally different kind of computing machine which is about processing instructions and is about having a processor available which successively or sequentially processes these instructions very fast and very exact .*
9. *this is not working for the parallel computing of the brain .*
10. *there we have multiple processors which run slowly , inaccurately , but pass the information in a massive parallel way in each case , that means , which are strongly linked with each other and also carry out computing performance in a parallel , joint way .*
11. *that means , you can talk about regions in the case of neural computing or in the brain , that means , we know that there are particular regions in the brain , which are responsible for image processing or for acoustical hearing or for speech .*
12. *however the computing is partly very much spread within these regions .*

13. *that means , you can 't usually pick one single nerve cell and say , this one is supposed to detect the third Fourier coefficient for the vowel A .*
14. *this isn 't possible .*
...
118. *in the case of the nerve cells it isn 't values I put in , but actually impulse frequencies , that means , how strongly a particular input fires .*
119. *that means , if I have inputs at a cell then I can reach a particular potential which leads to also firing accordingly at the output and hence analogically in this case to switch to on .*

120. *good .*
121. *the big open question is , of course , this all has its merits but who will give me these weights W ?*
...
140. *and formally said is if XI , that means , a feature I which belongs to class one , hence if this is labeled as class one if then the product with the weight vector is bigger than zero , and for features which belong to class two the product will be smaller than zero , then we will know that these features were correctly classified .*
141. *and , of course , the goal is to find a weight vector which is set in such way that it applies for all features hence that for all features of the class one the product is bigger than zero and for all features of class two the product is smaller than zero .*

142. *now , to make it even more simple for us we can say , all features of the class two where the product is supposed to be smaller than zero , as you know , these we are also putting on the other side now .*
143. *that means , now we have labeled data , labeled data for class one and labeled data for class two , and now we are going to take all labeled data , which belong to class two and attach a minus sign to them .*
...
513. *good .*
514. *don 't run away , please , I still have one slide .*
515. *or two .*

516. *what I still wanted to briefly tell you about this topic is two important points , please keep listening , two totally important points .*
517. *hence , we are dealing here with a non - linear classifier , which also is able to find curved separation lines , as you have seen here in this example .*
...
548. *however , also the additionally problematic processing of sequences .*
549. *that means , classification , an important step , prediction , an important step , but we ultimately also need a treatment of sequences of such objects , and this we will address next .*
550. *we will see you on Monday .*

Example Summary of Introduction Segment(5 out of 12 Sentences)

2. *today we want to get busy with the second lecture of machine learning , and we want to get busy here once again in somewhat more depth with a further class of learning algorithms , which all are collectively called neural networks , and we want to get familiar with one of the most popular approaches , the so - called backpropagation algorithm .*
3. *you can certainly also learn neuronal networks as a classifier for many classification tasks which we met already , which would belong in the group of non - parametric , non - linear classifiers , and thus is also offering a variation which represents non - parametric , non - linear classifiers , which thereby are of course especially interesting for the case of more complex , complicated classification tasks like those we have seen so far .*
5.*we rushed through this last time in a big hurry , so that I think that it would be certainly useful once again , if we would re - sketch it , and then would deal with the so - called multilayer perceptron or the neural networks by building up on this .*
10.*there we have multiple processors which run slowly , inaccurately , but pass the information in a massive parallel way in each case , that means , which are strongly linked with each other and also carry out computing performance in a parallel , joint way .*
11.*that means , you can talk about regions in the case of neural computing or in the brain , that means , we know that there are particular regions in the brain , which are responsible for image processing or for acoustical hearing or for speech .*

Example Summary of Content Segment includes (424 out of 495 Sentences)

13. *that means , you can 't usually pick one single nerve cell and say , this one is supposed to detect the third Fourier coefficient for the vowel A .*
14. *this isn 't possible .*
15. *we have a group of cells which cooperate there and which jointly recognize patterns or carry out computing performance in this way .*
16. *now , in this way we have a totally different kind of strength of these neural computing approaches , which are , of course , particularly good .*
18. *hence you only need to look at how athletes , for example , in gymnastics , let us say , are controlling a body and are able to manipulate it accordingly in a markedly impressive way .*
19. *we still are far away from this .*
20. *there are no robots which can do this in this way .*
21. *of course , there are experiments .*
22. *hence there are interesting approaches , indeed .*
23. *you have certainly seen the humanoid robots which are developed particularly in Japan .*
24. *this is certainly very impressive .*
25. *however , this is still behind of what a little child is already able to do after a few years , essentially .*
26. *it is , of course , an excellent way how the human brain does this .*
27. *on the other side it is also clear , of course , that there are particular tasks the brain can 't do well , that means , for example arithmetic , of course , we were already talking about extracting roots the last time .*
28. *of course , these are approaches which essentially work better by using a processor , which will successively execute program commands and program code , than with a distributed machine and with inaccurate processors .*
29. *now , consequently , this complete area of neural networks also had many names .*
30. *I only want to show this briefly , so that you will know what we are talking about if it will come up , and also know that people essentially very often mean the same thing with different accent .*
31. *partly , people are talking about these artificial neural networks .*
32. *connectionist models or plainly just connectionism , this is dealing with the complete computing methods of machines , which are trying to solve tasks using the linkage or connectionism of many computing elements .*
40. *hence , now we are talking about the working in parallel of many processors which have many interactions with each other , a comprehensive communication whereas the processors themselves don 't need to be exact but need to process widely distributed knowledge .*
41. *and the hardware is dedicated in this case .*
42. *that means , at least in the model , at least in the brain it is such that particular cells are solving particular tasks .*
43. *there you aren 't repeatedly loading new programs in the same cell but this is dedicated hardware .*

44. *because of this we also have very many of them .*
45. *that means , we have a whole brain full of cells at the size of billions , so that we have there , of course , for all possible tasks the corresponding computing performance .*
46. *now this , of course , doesn 't work by loading programs or by writing programs in this case , because this interconnecting and the interaction of so many elements is , of course , very complex , this wouldn 't be practicable at all .*
47. *hence most of the parallel computers are exploiting parallelism , of course , nowadays for tasks where the computing work for the different elements is relatively simple and similar , so that the same computing step gets distributed , so to speak .*
48. *hence depending on the parallel computing architecture which is used , this is the most popular one , of course .*
49. *hence , digital signal processing chips , for example , these are also distributing computing performance , but here these are the same computing steps on different processors in each case .*
50. *now here , if I would build such hardware then these would be different cells , by all means , which would do different things and would communicate with each other in a different way .*
51. *and this , of course , is very complex , this can 't be programmed anymore but in this case we totally critically need learning algorithms , of course , which learn this by using data or tasks .*
53. *how is this working ?*
54. *the advantage is , of course , we have simple computing elements so that this would certainly be appropriate for the parallel implementation .*
55. *however , this would also still be discussable why this is now not exactly like such , after fifteen years .*
56. *why aren 't there many neuro - computers which we would consistently employ day by day ?*
57. *there are reasons for this .*
58. *the uniformity or the simplicity of these computing elements is very attractive , because here I am able to work with very simple learning algorithms , and , of course , also because these computing elements are actually doing the same , no matter which signals I push in , I can also use this architecture for a complete facet of different tasks .*
59. *and this usage of many of such tasks is , of course , then also interesting again because then I will be also able to fuse all possible tings .*
60. *we know , that humans , for example , definitely mix or link signals of different modalities with each other or different knowledge sources in interesting ways which you wouldn 't expect at a first glance .*
61. *if you came to us to the lectures about neural networks for once then we would show you , for example , videos about lip reading .*
62. *that means , we know , for example , that humans unknowingly look at each others lips to complement the acoustic speech signal by the visual information of the lip movement .*
63. *hence there are very interesting effects where you , for example , would be able to study the fact , that you initiate the wrong auditory percept by playing the wrong video signal , so to speak .*

64. *that means , we would say , point to the box , and the video would play , point to the fox , and you would suddenly hear , point to the fox , hence what you had seen visually but not what you have heard .*
65. *that means , the perception is playing tricks on us or the brain is merging information which could definitely come from different sources . and behind this , of course , is such a neural architecture which essentially actually doesn 't care a bit where this signals come from , how they were presented , but which ultimately depends on if we can work with a percept eventually , with a good connection , or that we also can optimize it .*
66. *now , the fact that it is not conscious to us that we are doing this , points out again , of course , that we are only learning this by using it or by doing it .*
67. *every child plays with objects , plays with toys , and it wouldn 't suddenly have a class in the morning , today we talk about manipulation to the right and tomorrow it is going to be manipulation per se , but all of this happens playfully and all of this happens integrally .*
68. *that means , a child learns to use all of these signals in parallel and at once , so to speak , and such a computing architecture here , which is essentially able to use the facets of different signals , will definitely be the correct one , of course .*
69. *now , to start this , or at least to look at this , what people from computer science thought to be models , this entire area of neuro - informatics is nowadays meanwhile dealing with many sub - areas .*
70. *some of them , which are mainly concerned with biology , these in fact study cells in the nervous system , and try to understand what thus is happening in the nervous system , what is happening in the brain , to be able to classify this and to understand the processes accordingly .*
71. *separate from this there is the entire area of machine learning .*
72. *in the case of the task of machine learning , which we are actually also discussing during this lecture here now , it is less about to see what the brain exactly does with the cells , but it is more about to at least derive particular principles from the brain or from a brain - like computing method , I am saying this very carefully , which we could use to also build particular tasks in particular problems , practical tasks in the case of cognitive systems .*
73. *a further area certainly is the theory .*
80. *we since have spoken about this several times during the lectures , that we have a simple computing element which works with a decision function , G of X , and which then has the form of the scalar product of the input vector X multiplied by a weight vector W .*
81. *you need to think about this as an organism which would receive particular signals and , as said before , some time in the evolution or in the development of our intelligent beings it was certainly such that organisms then had tried to find fast decisions by using this switch element .*
84. *now , such an element has the input vector X , first of all .*
85. *that has features .*
86. *these features could be image pixels or acoustic coefficients and so on , and each of these features will be weighted , well , it is green , it has teeth , four little legs , it is long , and then the decision should be , dangerous , better run away .*
87. *well , and this decision doesn 't need to be an object , necessarily , it could be an action , it could work directly , but it should lead to a classification .*
88. *and it partly is very simple , of course , if it is a switch element , where I say , this is linearly decidable .*
89. *then I simply say , these are features , I weigh these features with the weight vector W and sum it up , and then will get a threshold .*

90. *and this threshold is shown on top here , namely , that I say , from a particular switch value I will switch to class A , running away , or to not running away , staying .*
91. *and thus I have my classification result .*
92. *now , this threshold is set to be zero on top , because it is not always the case that we would have thresholds which are not equal to zero for certain , we have this W zero at the bottom here , that it is always like this .*
93. *we shift the entire decision at W zero , so that we will be able to put in an arbitrary switch element in this vector space . okay .*
95. *now we want to see , how to learn such things .*
96. *we have said , as you know , I don 't want to have to program these weight vectors or I even can 't program these at all if it gets bigger and more complex . that means , we need a learning algorithm , which will do this .*
97. *first of all , we have discovered that this decision function GX also supplies me with the distance of the decision level at the end .*
98. *hence , I am able to show that this expression W zero plus the product of W and Z will deliver the distance of a feature vector of the decision level , after I have adjusted this weight vector , of course .*
99. *hence , if I would have two separable classes , I would train a good separation line , which sits in - between , then the normal vector on this separation line W will be the vector which I call weight vector and which also will define my separation line .*
100. *and if I would multiply this by the feature vector of an input date plus the threshold weight , then I would in fact receive not only the decision of whether I am on the right of the line or on the left , but I would also obtain the distance of this line .*
101. *and this is useful , of course , because this delivers in some respect also the distance or how strongly I am sitting from the separation line in the one or the other class in each case .*
102. *and when I do all of this my element looks like this , that means , graphically depicted , if I wanted to present this rather like a cell , hence until now we had it simply mathematically presented .*
103. *we also can see how the analogy to the Bayes rule is , that we hence have the depending class conditional and the a - priori probability .*
104. *here this is more such a representation of a system diagram , that means , if we wanted to draw such a switch element , then it would look like such , that I namely put in features down here , weigh each of this features , and this is going to be my scalar product , this is going to be my multiplication with the weights .*
105. *I sum up all of this and pass it into a threshold weight or in this , excuse me , not threshold weight , into the threshold , this is the decision function , of course , which in this simple case that we have already talked about , would hence be larger than zero , smaller than zero , this would be such a hard limiter , which says now the element is on or off .*

106. *but I also can build softer decision functions using such a threshold logic like here , where it has a particular transition , or a continuous transition like the Sigmoid function .*
107. *we will be interested in this further on , for the reason we will get to know in a minute , namely it is a continuous function , which is differentiable .*
108. *that means , I don 't need to make a hard decision here at any point , but it has a soft transition here as you can see pretty nicely .*
109. *however despite all of this , it is a non - linear function again here and a function which leads to a similar decision , namely that an element switches to on or off here .*
110. *now , the mathematical formulation if this Sigmoid function is such that the output YJ equals one divided by one plus E to the power of minus XJ whereas XJ is the input of this switch function in this case .*
111. *that means , if I put in a value here into this Sigmoid function then I receive the according switch value or output value of this Sigmoid function .*
112. *that means , I take now the sum of the weighted input vectors and lead them out of this switch element , and then receive the output of this Sigmoid function or of this switch element at the top here .*
113. *that means , my diagram essentially looks like such .*
114. *input features weights on each of this input features , I multiply all input by the weights , then I sum them up , I receive the weighted sum of the input features and lead this into the switch element then , which then switches to on or off .*
115. *this is analogous to what nerve cells do , in some regard .*
116. *in the case of the nerve cells it isn 't values I put in , but actually impulse frequencies , that means , how strongly a particular input fires .*
117. *that means , if I have inputs at a cell then I can reach a particular potential which leads to also firing accordingly at the output and hence analogically in this case to switch to on .*
118. *good .*
119. *the big open question is , of course , this all has its merits but who will give me these weights W ?*
120. *how will I adjust these weights to get a working perceptron , a functioning learning element .*
121. *we have said already we wouldn 't want to program it but automatically train it .*
122. *now , the switch element is depicted once again here , simply only with another notation .*
136. *good , how does this work now ?*
137. *if we have given this expression here for the first time , that we have it from zero to N , that means , we have the simple product here , and the task is now , we want to find this W , this weight vector .*

138. *then we certainly know , and this we have addressed last time already , that all feature vectors X will be correctly classified if it applies that the product of the weight vector and the feature vector lay on this side and the data which belong to the other side , there the product needs to be negative , then we know that we have classified correctly .*
139. *and formally said is if XI , that means , a feature I which belongs to class one , hence if this is labeled as class one if then the product with the weight vector is bigger than zero , and for features which belong to class two the product will be smaller than zero , then we will know that these features were correctly classified .*
140. *and , of course , the goal is to find a weight vector which is set in such way that it applies for all features hence that for all features of the class one the product is bigger than zero and for all features of class two the product is smaller than zero .*
142. *now , to make it even more simple for us we can say , all features of the class two where the product is supposed to be smaller than zero , as you know , these we are also putting on the other side now .*
143. *that means , now we have labeled data , labeled data for class one and labeled data for class two , and now we are going to take all labeled data , which belong to class two and attach a minus sign to them .*
144. *that means , we attach the minus sign to the data of class two .*
145. *and we know , of course , that the product is supposed to be negative in the case of a correct classification of the data of class two .*
146. *that means , for the case this data was correctly classified and we have attached the minus sign , hence all of this is also supposed to be positive , because I now have the data mirrored to the other side , so to speak , or have put it over there .*
147. *that means , if I simply write a minus sign in front of all data of class two , in my training data , then it applies that I should see the product positively for all correctly classified data .*
148. *that means , for all data , no matter if class one or two , it is supposed to apply that the product with the correctly classifying division vector is supposed to be positive .*
150. *no , around the zero point .*
151. *that means , we put all of this to the other side .*
152. *that means , we certainly want , that the product will become positive now .*
153. *and if I then simply write a minus sign in front of it , then it applies that the product changes from a negative product to a positive product in this case .*
154. *this we have exercised this already , as you know , that we pulled in the threshold weight already .*
157. *good , thus we now are able to develop the criterion function .*
158. *that means , in this case the criterion means now , we have said , as you know , that for all data which was correctly classified it needs to be positive , and now I want to build a penal criterion , so to speak , or such a criterion which tells me how bad my classifier still is , and this is , of course , all incorrectly classified data .*
159. *hence in my situation now , all incorrectly classified data would then be negative , of course .*

160. *that means , we just have said , a correctly classifying weight vector W would lead to the fact that all data would deliver a positive product , and for each date which is incorrectly classified now needs to apply now , that it delivers a negative product .*
161. *now because of this I can produce a criterion and say if I add all incorrectly classified data together then I get the sum of the negative products , of course , and if I once again attach a minus sign to these , then these will again be positive products , of course .*
162. *hence please excuse , if there are several minus signs coming after each other , but you see here , incorrectly classified tokens or incorrectly classified data would result in a negative product .*
163. *having the attached minus sign in this criterion function we will obtain a positive product again and the sum of these positive products will give me , so to speak , a measurement for the badness or the size of the problem , so to speak .*
164. *hence this is going to tell me how much I am still doing wrong or how big my cumulative error of all incorrectly classified data still is .*
165. *now , if this criterion is zero , that means , if the sum of all products of all incorrectly classified tokens with the weight vector , with an attached minus sign , resulted in a value of zero , then we will know that there aren 't any values left , or that there aren 't any feature vectors left which are incorrectly classified .*
166. *then we will be finished and will have found a solution vector which will deliver a separation line between the two classes .*
167. *now is the question , this would be the goal that we get this down to zero , as long as the perceptron is not correctly classified and the value is still positive .*
168. *that means , we want to get down from this positive value to the zero value .*
169. *and this we approach such that we say , we calculate the gradient and say , in which direction do I need to turn the weight vector then so that this criterion function JP becomes smaller and thus this criterion function JP will eventually tend toward zero .*
170. *now , we can calculate the gradient .*
171. *we simply can calculate the gradient from JP to W , we want to know , of course , how intensively we need to alter the weights so that JP becomes better , that means , we want to determine the gradient from JP to W .*
172. *and this we can calculate .*
173. *and this then has a simple solution , namely , that the gradient , calculated from JP to W is simply the sum over all negative feature vectors of the incorrectly classified tokens in each case .*
174. *hence we know now from this expression , from this criterion function , that the gradient isn 't anything else than adding together all incorrectly classified tokens with an attached minus sign , and this will represent my gradient .*
175. *now , what am I going to do with the weight ?*
176. *we have said already , we are calculating the gradient for the reason that want to know how I would want to adjust the weight so that it becomes better , so that this criterion becomes smaller . and then I will be also able to establish this update rule , which is simply telling me I want to deduct this deviation of the criterion function after the weight vector . and exactly this will give me the WK in a learning step K , where I add a value to the weight vector W which is proportional to the gradient which I have determined here .*
177. *and hence then I get to this update rule , at the learning step WK plus one my weight vectors will be set to the weight vector in the learning step before WK plus the gradient , that means , the sum of all incorrectly classified tokens provided with the weight or with the adjustment factor Rho .*

178. *so .*
179. *now this is , of course , a considerably simple solution , and amazingly simple , because what I actually only need to do in the case of the perceptron learning rule is I start with an arbitrary weight vector , see where this separation line appears , I search for all tokens which are incorrectly classified , literally simply add these together and provide them with a learning step Rho and simply add this to the existing weight vector .*
180. *hence I essentially take my given weight vector and simply add the incorrectly classified tokens .*
181. *now , this is of course a very , very simple learning rule and we want to study how this works step by step .*
182. *and then I am also going to show you a little applet to also demonstrate that it really works .*
183. *by the way , you can also prove that the algorithm will actually converge .*
193. *hence these are my four dots here and I want to train a perceptron that will separate these four dots from each other .*
200. *you can certainly imagine that it hence is possible to move this separation line a little bit up and down here .*
211. *that means , we will receive a criterion JP equal zero in this region .*
212. *and as soon as we are going to be outside of the region , as we have just addressed , particular vectors will be incorrectly classified , of course .*
213. *and this leads to the fact , that we will find the criterion JP as positive , that means , a positive value for JP which is , of course , proportional or as large as the sum of all incorrectly classified feature vectors .*
214. *and this leads to the fact that outside of the solution region , we are seeing the solution region down here with a criterion value of zero , as soon as we will be outside of this solution region we will , of course , have a criterion JP , which will start to become positive and to increase , and will become worse , of course , as further as we will move away from this solution region .*
215. *now we are also seeing here that it makes sense to build the gradient here , because we really want to drive down at this slope , so to speak , in the direction of the gradient until we will be in this solution region , and thus also will have a solution .*
216. *now for our example , what does this look like ?*
217. *that means , we would start with a random weight vector , for example , if we say , initially we would be here at the zero point and we would classify everything incorrectly , then , of course , the first weight vector we would find , would be the sum of all features X.*
219. *if everything was initially incorrectly classified , then I need to simply add all vectors of the data together .*
220. *and if we look again where all of them sit , that means , hence we have here these vectors , which all belong together , that means , we would add this one to this one and then here and there .*
221. *oops .*
222. *that means , we will obtain the sum as the first step , if we come before this zero point here .*
223. *that means , we make a step inside here , that means , please apologize , here are three examples in this case , to make it easier , three tokens , and if you added these three tokens together , then we would receive Y one , Y two and Y three , added together .*
224. *this is a vector Y two plus Y three which points at this point two , right ?*

225. *hence that means , if we added these three vectors Y one , two and three together we would end up at this point two .*
226. *now , this point three , if we imagined a vector , from the origin here to point two , then this separation line would be here , and everything at the above side of this separation line is correctly classified , hence Y two and Y two would be correctly classified , and Y three would be an incorrectly classified token in this case .*
227. *that means , I didn 't find a solution vector for my point two , because I still have Y three as an incorrectly classified example in my training data .*
228. *that means , I need to additionally add this feature vector Y three to this weight vector which points to two .*
229. *and I am doing this , and thus I will get to this point number three .*
230. *and now I am testing again , do I have a solution vector now ?*
231. *and we realize that the criterion isn 't zero still .*
232. *that means , we still have an error here , because in this case , of course , the normal vector would be at the point three here .*
233. *that means , my separation line appears like such .*
234. *and in this case , of course , I have incorrectly classified point Y one .*
235. *that means , I again need to add Y two , and then the solution vector or the weight vector is jumping to point four .*
236. *at point four we have again incorrectly classified the feature vector Y .*
237. *this is added again and then we will get to point five .*
239. *because all points are correctly classified in this case , my separation line is sitting down here and I am done .*
248. *hence here is our perceptron .*
249. *hence we basically have an arbitrarily chosen straight line here .*
250. *we can initialize this then with different weight vectors at the beginning .*
251. *hence this is just a random choice of weight vectors in this space , it is irrelevant .*
252. *and now we can start here for example to build a database of dots which we will label as class one .*
253. *and other dots which we will label as class two .*
254. *you see , that my random straight line in this case already classifies these dots here , the red dots on the right of the straight line as correct , the red dots on the left as wrong , and the blue dots as correct again because , of course , they belong to the other class .*
255. *now I can initialize this , arbitrarily as we have seen already , and hence can now start to check if the data is correctly classified in each case , like we are seeing here .*
256. *that means , I can go through these in each case and then also can apply the learning rule accordingly .*
257. *hence , if I would go for a complete round through the entire training set and would then add all of my weight vectors to do a learning step , then this would modify the separation line iteration after iteration .*
258. *you see , we could , for example , also switch on which of the predecessor steps the weight vector had .*
259. *that means , we have had the separation line here in such a way , have realized that particular data had been incorrectly classified consequently , and that then a new weight vector has been adapted here .*

260. *now , here we are now having a weight vector which separates the red ones and the blue ones , with the exception of these three errors .*
261. *that means , it is to be expected that only the separation line will drift to this direction where the errors have originated , because these are the tokens , of course , which I will now add to the weight vector during the next step .*
262. *now this is also happening in this way , hence I have these dots added , the weight vector is jumping to over here , and I am making another error again , which is in the other class . that means , I need to make a further step which will move everything back to the right direction .*
263. *and at that , in this case it still has been only a wrong token , so that my weight vector is only drifting a bit in the right direction and the algorithm is terminating .*
264. *that means , in this case we are done , and it doesn 't matter anymore how often I am going to let this continue .*
265. *we see we are sitting on a separation line here which separates the correct data from each other .*
266. *now , let us yet look at a few problematical cases .*
267. *here for example .*
269. *hence if we would let this run , for example , then we would realize that this data is , of course , beautifully linearly separable from each other .*
270. *that means , we have had a relatively easy task here , where the red ones were well separable from the blue ones , linearly separable by a linear separation line .*
271. *but you can now ask yourself , of course , now , was this really an especially good solution ?*
272. *maybe you let me repeat this once again , let us see what happens .*
274. *let us take these for example , for once .*
275. *is this a particularly good solution ? well , by purely looking at it , for us humans who would observe this , we would say , even if this is a solution for these training data , however , it isn 't the meaning of a classifier to only correctly classify the training data , but to also carry out a correct classification in the future in the case of a test when new data arrive .*
276. *and of course , here the question comes up , well , I as a designer would say here , this line is too drastically close to this red dot for me , so that this would actually be an inapt choice .*
277. *it would certainly be better if this line would lay somewhere here in between .*
278. *and this already is , for example , one of the problems in the perceptron algorithm , that it will , of course , eventually terminate , even though the separation line isn 't necessarily optimally set in a way which would have been perhaps picked differently in respect to the classification in the test case .*
279. *now , another problem which we have here , is , of course , in the case of non - linear separable problems .*
280. *well , we could play with these through the entire lecture , but you see already , the algorithm will never terminate in this case , because there is in fact no linear separation line which separates this data from each other in each case .*

281. *and , of course , it is a large problem that a linear classifier can 't solve a non - linear problem either .*
282. *that means , here we need something powerful , or at least a slick compromise .*
283. *also the perceptron algorithm actually is relatively inept in this direction because it jumps here and there with the respective wrong tokens .*
284. *that means , if I do something like this here additionally , for example , then it will , of course , totally wildly jump around here and there , because my incorrectly classified data are partly sitting outside here , of course .*
285. *now , what would a person do ?*
286. *if I would force you and would say , you have to build a linear classifier here , then you would say , okay , so in the case of these dots outside there , the red one and the blue one , I have bad cards .*
287. *I am going to put something in here somewhere , which is at least going to be a good compromise .*
288. *and also here the perceptron algorithm actually isn 't very appropriate , because despite of this it will still simply widely jump around .*
289. *and then I as a designer would somehow need to decide when I am actually going to terminate .*
290. *that means , I could stop in a particularly bad situation .*
291. *hence , if I would do this here , would decide here , for example , I will simply stop at the iteration number such and such , then it could be , of course , that I will have identified a particularly inept solution here .*
293. *what can I do ?*
294. *well , the learning rate is a problem , of course .*
295. *I also wanted to totally briefly show you this , perhaps .*
296. *you could , of course , adjust this Rho , yes , this Rho , which we got to know a few minutes ago , is certainly not only adding these incorrectly classified vectors , but we can once again weight this move from the weight vector to a learning step .*
297. *that means , we can say , we add a factor once again which will tell me how fast I actually want to move this separation line from the one position to the other position .*
298. *and now you are seeing , of course , on the first glance why this could be good or bad .*
299. *hence , if I would pick this too small , then I could possibly endlessly exhaust this until I would finally find something reasonable , whereas I could want to solve the problem too aggressively in the other case .*
300. *if we would do this here very briefly once again , for example , in the linear separable case , and I will have this situation here , then this will possibly take too much time for me .*
301. *if I would pick bigger factors here , then it will jump faster .*
302. *that means , I will possibly get to a faster solution , but this would be also dangerous , of course , because the separation line is jumping around brutally , particularly if I would assume that I am having linear , non - separable data then I will get such wild jumping around so that the terminating afterwards at the end will be very critical .*
304. *because if I would terminate at the wrong position then I will get a particularly bad solution , of course .*
305. *that means , usually it is important , of course , to practically , empirically determine how to switch on this learning rate .*

306. *how fast should I make it ?*
307. *if I made it too slow then it is going to take me too much time .*
308. *if I learned too fast , then I am getting a pretty wild behavior , where I won 't be able to tell what is happening when the algorithm terminates , or when I need to stop .*
309. *initial weights , these are a further problem .*
310. *how would I actually start ?*
311. *which weights would I pick at the beginning ?*
312. *of course , here it can also be , that I would start with a weight vector somehow , which is totally far away .*
313. *this isn 't a problem necessarily in the case of the perceptron algorithm but we will also see , that this initial weight vector also plays an important role for the neural network or for the multilayer perceptrons .*
314. *now , problems , what would I do if the data isn 't linearly separable ?*
315. *or , as we have just seen , if the data is in fact separable , but I would have the choice between several possible decision levels or straight lines , which of these should I pick then ?*
316. *and as said before if data is in fact separable , but is non - linearly separable , what could I do then ?*
317. *now , during the entire work on the perceptrons . . . hence this was very much worked on during the sixties , there was much interest in the perceptrons . hence there perceptrons were also invented which have at least slightly put these problems , which we have just discussed , into perspective , or have slightly improved them .*
318. *here you can think of such relaxation procedures , for example , where I won 't simply only add the incorrect data together , at least , as an error function , as a criterion , and thus will get relatively hard jumping around , but will define a flat criterion function .*
320. *and one of these would be the squaring of this product , for example , to obtain a smoother surface which thus would offer a more elegant learning where I would then experience a better convergence , consequently .*
321. *a further trick I could apply here , is that I would additionally introduce a so - called margin B in this relaxation procedure , this is a certain margin value .*
322. *I am requiring , so to speak , in my classification algorithm , that the algorithm will have to continue for such a long time that it won 't just correctly classify the data in my training data , but also a certain security zone around the data .*
323. *you can see this here in the picture above in such way , that we are having the solution regions here , for example , where all data is correctly classified now , of course , but I do not want to have , of course , that a solution is directly sitting like such at the margin of my training data , but I do yet want to have a certain security zone around it .*
324. *and I can do this by defining this security zone B or margin B , and demand , that all solution regions have to sit in this solution region plus the margin B .*
325. *that means , the algorithm has to continue at most until I am really inside this security zone .*
326. *that means , I will have a certain distance from the actual margins , and then I will be able to tell that my solution here is perhaps more reasonable than another which sits directly here at the borderline .*
327. *still another solution would be that I used the mean square error as the criterion , and then in fact additionally used the distance to the separation lines of my data .*

328. *of course , the idea behind this is , that I say , it is not just about finding a solution where all data will have been correctly classified , but I additionally want to somehow optimize the distance of my data to this separation line , of course , to obtain a solution where the separation line is as far away from my data as possible , hopefully , and is , let us say , in a position which is more or less in the middle between my data .*
329. *now , the perceptrons , an awesome thing , much was written about this and much was published during these years .*
330. *and some when in the seventies all of this was crushed down by the beginning of the artificial intelligence .*
331. *especially from a book by Marvin Minsky , who wrote a book about the perceptron , and had actually shown there , that this perceptron is not able to compute very , very simple functions , and thus is simply not powerful enough to solve important tasks in the perception or in all of the problems of the artificial intelligence .*
332. *and this simplest function which thus became famous and also was discussed there , is , of course , the ExOr - function , where it can be shown that the perceptron is not able to do this .*
333. *ExOr is the logical function , of course . that means , I have two inputs , one is on , one is off , then the output should be switched on , and if both are on or both are off , then the output should be switched off .*
334. *you can think of this as feature vector space in such way that I would draw the two input tokens or the input characters one here in my feature vector space which I insert here in my ExOr - function , and say , if both input features are switched to zero , then my output shall be zero .*
335. *if both are switched to one , my output shall be zero , too .*
336. *and if one of both is at one and the other at zero , then I want to switch the output to one , this would be ExOr .*
337. *and this ExOr - function can 't be solved by a perceptron , because we have the situation here , of course , that a simple linear separation line between these two classes won 't be able to separate the classes from each other .*
338. *that means , we have a classical non - linear problem here , namely , that a separation line here . . . you can imagine , of course , here the separation line would certainly always jump around , and would never find a solution vector , because the data simply isn 't linearly separable from each other .*
339. *now , this can be proven , this can be shown , but it is already obvious from this chart here , that it won 't be possible by using a simple linear separation line .*
340. *that means , I need something more powerful , or at least several switch elements for once , which will do this .*
341. *now , at these days people figured that they would be able to build a multilayer machine .*
342. *that means , I can put together a machine from several of such switch elements , and can say , my first switch element , for example , is separating this dot down here from the other three , and then I am going to take these three here , and will train a classifier , which will separate these three dots from each other . this would assumingly enter a second line here .*
343. *and then , of course , I could solve the problem , however , I will need two switch elements right away , of course , which I respectively connect consecutively .*
344. *and then the question is coming up , I could partly certainly do this manually in such a simple task such as ExOr , so that I would separate the data as a start , and say , okay , I am initially having this linear problem , and then I will have that linear problem , and then I am going to consecutively connect these in each case .*

345. *this was maybe still working for this case , but now think of more complex tasks , where you can 't even really draw the feature vector space .*
346. *then we will have the same problem , as we have just discussed in the case of the unsupervised algorithms , namely , I don 't know at all the structure of my data , and I don 't know , how I would need to partition it now , to be able to establish an optimal classification performance by using a multilayered machine .*
347. *now , the idea of multilayered machines was in fact much discussed then , it didn 't , however , actually result in this practical breakthrough generally , because we thus can 't assure , of course , that we will receive a multilayer machine which will be optimized all together , or which also will minimize the error all together , or will optimize the performance .*
348. *hence , we need some type of algorithm , which will be able to train a connected multilayer machine all together .*
349. *hence what we need is an algorithm , which says , I have several layers , and it isn 't relevant at all what the single units do inside there , but I only want to see , that something correct comes out right at the top somehow .*
350. *this is like , if you have stocks in a company , then you won 't be interested in what the third employee is doing during his seventh shift in the big company , but you do want that the company ultimately makes a good profit .*
351. *and then you have a boss at the top , and he will need to seek now that his people beneath are doing the right thing , and so on and so forth .*
352. *you are wanting the total thing , so to speak , the total organism , optimizing the total system .*
353. *and what you will be interested in is the total output of this ensemble , or the total output of a multilayered machine , a bigger configuration of the single elements .*
354. *and we want to see , that each of these elements is doing exactly what is necessary , so that the total output will be optimized .*
355. *and this basically is already the idea for the backpropagation , this algorithm which is very popular .*
356. *and hence , this meanwhile is standard repertoire .*
357. *you will be able to actually already find it in Matlab or in all of your software packages .*
358. *you can also program it for once .*
359. *it is relatively simple to program .*
360. *this is a relatively short program , and does pretty interesting things through which you can in fact get up to a lot of things in pattern recognition .*
361. *now , a multilayered machine looks such , or a multilayered perceptron , I should say , looks such that I connect several of those perceptrons together and on top of each other .*
362. *that means , here once again I have the perceptron , like I got to know it before .*
363. *hence , each of these knots is a perceptron in each case .*
364. *that means , here I still have an input level and a perceptron , which switches to on or off , accordingly , but the output of this perceptron is not directly used as output , but is only put into a further level of switch elements , so to speak , which are laying above . and for this entire arrangement , it isn 't interesting for me anymore , what this switch element is actually doing down here but I have particular requirements in the output neurons , in the output switch elements on top of it .*
365. *and remember , this would be the situation in the case of my ExOr - problem .*

366. *I want the ExOr - function to be correctly computed , and if I have several levels of neurons beneath it , you will need to see what you will need to do , so that the complete solution will be correct .*
368. *can we sketch an algorithm which will do this for us ?*
369. *the backpropagation algorithm is going to do this .*
370. *the idea and the sketch is basically analogous to what people did in the case of the perceptron .*
371. *hence , remember , in the case of the perceptron we have said , we want to shift the separation line in such way that a certain criterion function will become better , step by step .*
372. *that means , we are going to calculate the gradient of this error function .*
373. *remember , our criterion function was the error function for the perceptron rule .*
374. *how bad is the output going to be , how problematic is the output going to be ?*
375. *and then we have used this error function or this measurement of the badness of the classifier , and have derived it according to the weights in the perceptron , to then adjust the weights in a way , that my result would become better during the next calculation step .*
376. *bye .*
377. *now , how would I do this now , if I had a situation where I wouldn 't have a single perceptron anymore beneath it but where I would have a level of several perceptrons which would be laying on top of each other ?*
378. *here the thing is not as simple anymore , because I can 't just adjust the weights beneath it , but what is important now , is that I adjust all weights in such way that the result is becoming correct at the top .*
379. *and the dependency of the output of the weights in the lower level won 't be directly determinable anymore or directly computable as a function . that means , the output won 't directly be a function of the weights in the first level anymore , but , remeber , the output will be a function of a function of a function of a function of the weights in the first level .*
381. *hence the output up here is certainly influenced in this case , for example , by this weight down here , but not directly in the form of a function , but the output here is the function of the Sigmoid function above here .*
382. *and this in turn is the function of the scalar product of the weights in the upper level .*
383. *and this , in turn , then is the function of the knot beneath it and so on and so forth .*
384. *now , if I wanted to know how I would need to adjust these weights down here so that the output up here will become better , then I would , of course , need to come down into these weights in the differentiation of my criterion or of my error function , and this I need to also propagate through , of course , and hence the name backpropagation through the different levels .*
385. *now , since the upper output is the function of a function of a function of a function of my weights on the bottom , I would need to know somehow , of course , how I am going to differentiate here mathematically .*
386. *and here we still remember grade school , that there must have been something , to let me know how to differentiate in respect to a variable , if I shift this variable with the output through several functions , and this is the so - called chain rule .*

387. *that means , if the output is the function of a function of a function of a function , then the derivation in respect to this variable which lays beneath is , of course , the derivation of the respective function , multiplied by each other .*
388. *and this basically already is the whole trick and the whole magic in this backpropagation .*
389. *that means , I am searching for the derivation of the error function DE in respect to all weights in the entire network .*
390. *and these I can compute such that I am going through the entire network by using the backpropagation algorithm , and will apply the chain rule in each case .*
391. *now , to do this very quickly , we see , that thus such an element now is looking such here , that would have an input feature YI , the weight with WI , WJ , would sum up all of these weighted inputs , and would push them through my threshold function .*
392. *here now is , of course , very crucial that we are going to work with the Sigmoid function , or very benefiting , and not with a hard decision function , because remember , we have already said before , the beauty with it is , that we can differentiate it , and exactly for this reason Sigmoid functions are used , because this , of course , simplifies everything .*
393. *that means , to be able to apply the chain rule , I need to be able to individually differentiate all of these functions now .*
394. *that means , the Sigmoid function up here has this form , and , of course , I can also differentiate it , and this I will need , of course , when using the chain rule .*
395. *now , we know what these elements look like .*
396. *this function calculates the scalar product , as we have seen , this function calculates the Sigmoid function , and at the very top at the output we calculate the error function between an output vector , which we will in fact observe and the desired output vector .*
397. *hence , also here we again have desired output results .*
398. *that means , of course , this would be in our classification , that maybe all data which belong to the one class would be classified as one , the others as zero , or whatsoever .*
399. *and then we will calculate the error between the output value which my network is computing and the desired value at the output .*
400. *hence , I remind you once again , if you would simply randomly apply random weights in the beginning , and are now going to run through the network simply with a input feature in forward pass , you would , of course , obtain any kind of values at the output which deliver any random result .*
401. *that means , now I can certainly carry out all of these multiplications , and can calculate my Sigmoid and so on , and then I will obtain a Y value at the end of this network which will , of course , however not yet deliver the desired feature or the desired performance .*
402. *now I am going to calculate the error .*
403. *that means , we are thus sitting up here at the top , and are first putting an input pattern in .*
404. *we have random weights .*
405. *we multiply these by the inputs , we calculate this threshold function , the Sigmoid function , now we are again passing the output of this into further random weights , and then will calculate the output activation of the output , then we will get any values which still aren 't something , and we will now calculate the error to the desired output target vector .*

406. *that means , how would I like to have these output neurons to switch , and how do they switch in fact ?*
407. *and then I calculate the error between the desired D , the target vector and the actual output vector YY .*
408. *good .*
409. *this is my error function .*
410. *this I want to differentiate now in respect to the weights in the entire network .*
411. *now I need to apply the chain rule .*
412. *first of all we need to see how we build the derivation of the Sigmoid function .*
413. *that means , we have a Sigmoid function YJ , because we need this , of course ' if we are going to apply the chain rule now .*
414. *YJ is one plus one plus I to the power of minus XJ .*
415. *I don 't think I need to redo all of this here now .*
416. *you may look at it in the slides , they are on the web .*
417. *but if I am differentiating now the output of the Sigmoid function YJ with respect to XJ , then this is going to be school mathematics , and I get an expression which I can form in a pretty simple way by using YJ .*
418. *that means , if I resolve this whole thing in respect to YJ then I will get an expression as differentiation , as derivation of YJ with respect to DXJ , which is relatively simple , namely YJ multiplied by one minus YJ .*
419. *this is my derivation of Y with respect to XJ .*
420. *now , if I know this , then I thus know that this is my error function , this is my Sigmoid function , my weighted input features are looking like this .*
421. *then I can get from this top level to the level beneath it by deriving the output error with respect to YJ , first of all .*
422. *that means , I am building the differentiation here , I yet first of all derive E with respect to YJ , and this is giving me YJ minus DJ .*
423. *and thus I calculated the derivation of this function .*
424. *now I need to calculate the derivative of this function multiplied by this first derivative .*
425. *that means , I am taking this D with respect to DYJ which I have already calculated , and need to now calculate DY with respect to DXJ .*
426. *and this is the derivative of the Sigmoid function , DY with respect to DXJ .*
427. *that means , this DE with respect to DYJ , of course , is my error function , and DYJ derived with respect to DXJ is the derived Sigmoid function . we have just seen what this is , this is YJ multiplied by one minus YJ .*
428. *and now I need to go a step beneath and say , what is it now going to be for the next step ?*
429. *that means , meanwhile we know what DE derived with respect to DXJ is , and in the next step I will need the derivative of this sum of the weight parameter .*
430. *that means , I am going to derive DE with respect to DWYIJ .*
431. *that means , I am going into this next step , and this is once again DE derived with respect to DXJ , and then now in the next step DXJ derived with respect to DWIJ .*
432. *and this we see according to this expression up here , will simply be YJ .*
433. *and to continue later on , to get into the next level , I will , of course , also need the derivative of DYJ with respect to DYI .*

434. *that means , the derivative of the output of this neuron with respect to this neuron , because this I will need once again in the chain rule , to be able to get to the next level afterwards .*
435. *because let us put it this way , if I would have only one level , I would be able to stop here , because I only need the derivation with respect to the weights . but later on I will also need the derivation of the weights in the next level . that means , to be able to apply the chain rule , I will then also need here the derivative of the output with respect to the input of this neuron .*
436. *this I will be able to do here , too .*
437. *that means , also here , the chain rule again .*
438. *I obtain DE with respect to DW , excuse me , DE with respect to XJ , multiplied by DXJ with respect to DYI .*
439. *that means , this is the derivative up until here , and then the derivative with respect to YI , once again .*
440. *and now we will have a sum inside here .*
441. *and the question is , why is there a sum inside here ?*
442. *because , of course , the derivative after the input in this neuron is not solely depending on the output of this neuron , but also on the output of the other neurons , which we had above , right ?*
443. *that means , you need to remember , that there is not just one neuron sitting here in the top level , but several which are connected with this input value YI .*
444. *that means , of course , I need to also consider the contribution of the derivative of all of these output values into this input value , and consequently also calculate this sum .*
445. *so , that was it , basically .*
446. *thus we know , how we get down to the input of this neuron , using the chain rule .*
447. *and if we know how this input of this neuron needs to change so that the output improves . . . by applying the chain rule I am , of course , able to carry the game to the next level , and say , how then would I need to consequently alter the levels underneath , so that thus the output will change ?*
448. *that is why the algorithm is called backpropagation .*
449. *I compute only once in forward pass , calculate my error at the output , and are now successively applying the chain rule , from level to level to level , and then will have the complete gradient according to all of these functions .*
450. *and now , of course , I can read off the DE with respect of DWIJ here in each case , from each of these levels .*
451. *that means , I am receiving all of the derivatives of the error with respect to all weights in my entire network , not just at the output level , but also from the level underneath , or from several levels .*
452. *hence I certainly also can build such multilayer perceptrons , and hence also carry out the derivation there .*
453. *very briefly again something about the statistical interpretation .*
454. *unfortunately I won 't be able to demonstrate or prove this here now , but there are theoretical papers about this .*
455. *the question is , if I have trained such a backpropagation net , what does the output actually mean then ?*
456. *this output is actually a switch element , which is trained to one or zero , maybe a classification task , but would I be able to interpret this somehow ?*

457. *if we remember the Bayes rule once again , would this be anyhow a statistical interpretation in any sense ?*
458. *the answer is , yes .*
459. *you can show , that if I trained such a net with binary target vectors . . . hence , if I define , for example , a classification task in a backpropagation network , I would set up a net where the output neurons switch to a class in each case .*
460. *that means , if I had a three class problem , then I could set up such a network , that could be maybe totally connected here , and then I could say , if this would be class one or the class A , let us say , this here is ABC , then I could say , I will have a vector as a target vector which would say one , zero , zero , if a date would belong to the class A , and if it would belong to the class B , I would get a target vector which would say zero , one , zero .*
461. *that means , I would train my net in such a way that one of the neurons will fire for one of the classes in each case .*
462. *this is a totally typical application of a backpropagation net for classification .*
463. *in this case then , I would , so to speak , aim for a binary vector as a target vector .*
469. *I can now use such networks to solve classification tasks .*
470. *and there I would like to still quickly show you an applet , how it will look as a classifier . okay .*
471. *okay , let us assume we had the ExOr - problem still at the very beginning here .*
473. *now , this won 't work with the perceptron , as we have already seen .*
474. *if I would now use a totally simple multilayer perceptron , in this case it would be solvable .*
475. *first I could begin to train a backpropagation network , such as it is depicted here .*
476. *we are seeing the output of this network up here .*
477. *now we can . . . oops , that wasn 't how it was thought to be , of course .*
478. *we see here after ten thousand steps . . . that means , what we are doing here , if we calculate the gradient , we are getting now a new gradient in each step , which tells me , how I need to adjust the weights in the entire network so that my error improves .*
479. *the learning rule , this I didn 't show so far , unfortunately , is basically totally analogous to the case of the perceptron , namely , that I use all weights , and add to them this gradient .*
480. *that means , I take , so to speak , a step in the direction of the gradient .*
481. *remember , my gradient tells me , where to go , so that it will get better , and I am now taking a step into the direction where it will get better , where my criterion , my error will improve .*
482. *and thus I am having different learning steps .*
483. *we will see this then in great detail during the lecture about neural networks , how I would adjust these learning steps and so on .*
484. *but if these would be correctly adjusted , then I am going to take steps at this hillside , walk down this hillside of the error function into the direction of the gradient , and try to find a better solution again and again .*
485. *and with this new solution I will then hopefully also get better classification outputs again and again .*

486. *that means , if I had , for example , passed through ten thousand steps here , I would see already , that I have a separation between red and green here , which would be already better for once in any case .*
487. *hence we here have three of such dots already correctly classified .*
488. *now I am going to continue here .*
489. *and you see , after another ten thousand steps , twenty thousand steps I am actually getting a solution , which is already finished .*
490. *that means , I have in fact solved the ExOr - problem here by using a multi-layered perceptron , where I am hence optimizing several levels of such perceptrons to solve this entire problem in an accordingly correct way .*
491. *now we would like to perhaps look at another example quickly .*
492. *if I had , for example , such a classification task here , where the dots are spread , they have a similar shape as ExOr , that means , it isn 't linear .*
493. *and we could initially also see here how this looks like , if we passed through a certain number , ten thousand steps .*
494. *you see , these separation lines are getting closer to a solution , but it still isn 't a solution .*
495. *we see , that it still is non - linear , because I also need limits at both sides of the green area .*
496. *but you see , that this green area can 't be totally linear either .*
497. *that means , you actually should also additionally find a curve .*
498. *that means , we are continuing here until we find according solutions .*

Example Summary of Conclusion Segment includes (11 out of 35 Sentences)

517. *hence , we are dealing here with a non - linear classifier , which also is able to find curved separation lines , as you have seen here in this example .*
519. *that means , if you would build a totally complex , humongous neural network with many neurons , nothing is h actually olding the network back from also building such a fence around each date , and thus , of course , it wouldn 't be a classifier which is generalizes well .*
521. *this entire problematic of the ability to generalize and of the complexity of a classifier is an important field , which machine learning is dealing with , and which we will also address then during the lecture about neural networks .*
522. *the other thing , what I still also wanted to say which is very important is , that such a neural network can not only be employed as a classifier .*
523. *here we have seen , how we , for example , calculate outputs for a classifier , where we say , we train the network , so that it switches to the correct class in each case .*
524. *but it is irrelevant what I require as an output from the network .*
527. *the stock market of tomorrow won 't be one or zero , it will have a particular value , or it will have a particular dimension .*
529. *I would possibly simply try to predict how the stock market of tomorrow will look like , given particular observations of today , then I could try to train such networks as predictors .*
533. *if you wanted to build a very complex function for any mathematical tasks , then you could certainly build a neural network , which wouldn 't calculate this function , though , but would simulate this function which would , so to speak , display exactly the behavior you would like to see from your function .*
537. *if you wanted to say , I want to access a particular point in the robot in three - D , and say , this is my target vector , and hence now I want to determine which streams or which joints and which angle do I have to apply to my robot arm so that the top of my arm will arrive up here .*
538. *a child needs to do something similar , because a child won 't be told how to calculate inverse kinematics with a lot of matrices and multiplications , however , it does this by frequent trial and error , it will discover sometime by grabbing things , that if I would see a point here , which I would like to access , I would need to do something with my little muscle here , so that my finger tips will arrive here .*
540. *this is really the last slide you will see , if you would come to the lecture about neural networks the next time , such an example is , for example , such a function approximator , which learns , for example , to drive a car .*