FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Using deep neural networks for sentiment analysis from utterances
Student:	Bc. Jiří Kožusznik
Supervisor:	prof. Ing. RNDr. Martin Holeňa, CSc.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2018/19

Instructions

1. Get acquainted with important kinds of deep neural networks and with their implementations in the developmant environment Matlab, including the possibility to use those implementations on GPUs and in the virtual cloud Metacentrum.
2. Get acquainted with the area of sentiment analysis, paying attention in particular to sentiment analysis from utterances, including the processing of utterances in Matlab.
3. Train at least 2 kinds of deep neural networks, each in several configurations, to classify the emotion of utterances in the public database EmoDB.
4. Compare the accuracy of test data classification between networks trained using the raw energy coding of the utterances and those trained using the MPEG-7 coding.
5. Compare the obtained results with published results of EmoDB classification by SVM.
6. Experiment with teams of classifiers including some of the considered networks and possibly also other kinds of classifiers.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 17, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Using deep neural networks for sentiment analysis from utterances

Bc. Jiří Kožusznik

Department of Applied Mathematics
Supervisor: prof. Ing. RNDr. Martin Holeňa, CSc.

May 9, 2019

Acknowledgements

I would like to thank prof. Ing. RNDr. Martin Holeňa, CSc for his help and support during our consultations, my family and friends for support during writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 9, 2019 . . . .

Citation of this thesis

Kožusznik, Jiří. *Using deep neural networks for sentiment analysis from utterances*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce zabývá problémem analýzy sentimentu z audio souborů, k čemuž využívá LSTM sítí, které porovnává se stávajícími klasifikačními metodami. Je navženo a implementováno několik postupů, jejich výsledky jsou v práci shrnuty.

Klíčová slova Analýza sentimentu, audio, LSTM, EmoDB, SDT, klasifikace

Abstract

This thesis deals with the problem of sentiment analysis from utterances by using LSTM networks. These are compared with some more widespread classification methods. Several approaches are proposed, implemented and compared to each other. The results are summarized.

Keywords Sentiment analysis, audio, LSTM, EmoDB, SDT, classification

Contents

Introduction **1**

1 Basic terms and definitions **3**
- 1.1 Audio-visual content . . . 3
- 1.2 Bark scale . . . 3
- 1.3 Audio Descriptors . . . 3
- 1.4 Classifiers and Classes . . . 7
- 1.5 Measures of Classifier Performance . . . 9
- 1.6 Employed Classification Methods . . . 13
- 1.7 Verification methods . . . 21

2 Approach **25**
- 2.1 Available tools . . . 25
- 2.2 Workflow . . . 26

3 Implementation **31**
- 3.1 Prerequisites . . . 31
- 3.2 Python . . . 31
- 3.3 Matlab . . . 32

4 Testing **37**
- 4.1 Berlin Database of Emotional Speech . . . 37
- 4.2 Experimental Testing . . . 37
- 4.3 Evaluation of results . . . 45

Conclusion **49**

Bibliography **51**

A Acronyms **55**

B Contents of CD **57**

List of Figures

1.1 An MPEG-7 architecture requirement is that description must be separated from the audiovisual content. On the other hand, there must be a relation between the content and description. Thus the description is multiplexed with the content itself.[1] 4
1.2 In figure A, the target labels can be separated with a line. This is called linear separability and can be done not only by SVM, but by any classifier whose output is a linear combination of input features. The figure B shows data which cannot be linearly separated, at least not in 2D space. As a linear classifier, SVM achieves the non-linear classification by adding another dimension to the data and separating them in the higher dimension. In the figure B, the data would be moved to 3D space and linearly separated by a plane.[2] 14
1.3 The difference in the information flow between a RNN and a Feed-Forward Neural Network. [3] 19
1.4 Diagram of k-fold cross-validation with $k = 4$. [4] 21

2.1 An overview of the sentiment analysis process. 26
2.2 An overview of the Friedman test process. 27
2.3 An overview of the sentiment analysis by LSTM network process. . 28
2.4 An overview of the sentiment analysis by LSTM network process. . 29

4.3 Compare accuracy MPEG-7 groups. 42
4.1 ROC curve for all emotions on the whole Berlin database 46
4.2 ROC curve for all emotions on the whole Berlin database 47
4.4 ROC curve for LSTM with 1 hidden layer with 250 neurons on the whole Berlin database 48
4.5 ROC curve for LSTM with 2 hidden layer with 250-100 neurons on the whole Berlin database 48

List of Tables

1.1 Confusion matrix in binary classification 12

4.1 Accuracy and area under curve (AUC) of the implemented classifiers on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest 38

4.2 Comparison between pairs of implemented classifiers with respect to accuracy, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the accuracy of the row classifier was higher : on how many parts the accuracy of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal accuracy of all classifiers (significance level 5%), the post-hoc test according to [5, 6] rejects the hypothesis of equal accuracy of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [7] 39

4.3 Comparison between pairs of implemented classifiers with respect to the AUC averaged over all 7 emotions, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the AUC of the row classifier was higher : on how many parts the AUC of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal AUC of all classifiers (significance level 5%), the post-hoc test according to [5, 6] rejects the hypothesis of equal AUC of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [7] 40

4.4 Accuracy of the LSTM network on MPEG-7 groups 43

4.5 Accuracy and area under curve (AUC) of the LSTM with 1 hidden layer on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest 43
4.6 Accuracy and area under curve (AUC) of the LSTM with 2 hidden layers on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest 44

Introduction

The recognition of emotional states in speech is expected to play an increasingly important role in applications such as media retrieval systems, car management systems, call center applications, personal assistants and the like. In many languages, it is common that the meaning of spoken words changes depending on speakers emotions, and consequently the emotional information is important in order to understand the intended meaning. Emotional Speech recognition is a complicated process. Its performance heavily relies on the extraction and selection of features related to the emotional state of the speaker in the audio signal of an utterance. For most of them, the methodology has already been implemented, and they have been experimentally tested and compared to Berlin database of emotional speech.

In the thesis, we use MPEG-7 low level audio descriptors [8] as features for the recognition of emotional categories. To this end, we elaborate a methodology of combining MPEG-7 with several important kinds of classifiers. For most of them, the methodology has already been implemented and tested with the publicly available Berlin Database of Emotional Speech [9].

Due to the importance of recognizing emotional states in speech, research into sentiment analysis from utterances has been emerging during recent years. We are aware of 3 publications reporting research with the same database of emotional utterances as we used – the Berlin Database of Emotional Speech, used in our research. Let us recall each of them.

The research most similar to ours has been reported in [10], where the authors also used MPEG-7 descriptors for sentiment analysis from utterance. However, they used only scalar MPEG-7 descriptors or scalars derived with time-series descriptors using the software tools Sound Description Toolbox [11] and MPEG-7 Audio Reference Software Toolkit [12], whereas we are implementing also a long-short-term memory network that will use the time series directly. They also used only one classifier in their experiments, a combination of a radial basis function network and a support vector machine.

In [13], emotions are recognized using pitch and prosody features, which are mostly in time domain. Also in that paper, the experiments were performed, and the authors used only one classifier, this time a support vector machine (SVM).

The authors of [14] proposed a set of new 68 features, such as some based on harmonic frequencies or the Zipf distribution, for better speech emotion recognition. This set of features is used in a multi-stage classification. When performing the sentiment analysis of the Berlin Database, the utterance classification to the considered emotional categories was preceded with a gender classification of the speakers, and the gender of the speaker was subsequently used as an additional feature for the classification of the utterances.

In the first chapter 1 important basic terms and definitions related to datamining and machine learning are described. These are needed for understanding of the thesis.

The second chapter 2 deals with some suitable tools for audio descriptors extraction and also introduces the proposal of several algorithms that can be used for sentiment analysis from utterances.

The third chapter 3 covers the aspects of practical implementation of algorithms that were introduced in the previous chapter.

In the last chapter 4 the implemented algorithms are tested on the real dataset and the results are compared and visualized.

CHAPTER 1

Basic terms and definitions

1.1 Audio-visual content

Audio visual content equals high quality, useful information with the target to present a story for the purpose of soliciting emotion or engagement. Audio-visual content can be presented in many ways: textual, graphical, in the form of video/audio etc.[15]

1.2 Bark scale

Bark scale is used to measure sound frequencies. Distances on this scale are perceptually equal for the human ear. For this reason the scale values are more and more linear below 500 Hz. Above the 500 Hz point the scale is almost similar to a logaritmic frequency axis. [16]

1.3 Audio Descriptors

Audio descriptors represent certain characteristics of an audio recording, either general, or deliberately chosen for a given purpose. They are obtained via digital signal processing (peak), or by classification (music genre). The levels of audio descriptors refer to the length of the described audio segment, where low–level descriptors consider an instantaneous feature, mid–level descriptors describe a particular interval and high–level descriptors describe the whole file. The characterization of audio descriptors is taken over from [17].

1.3.1 MPEG-7 Audio

MPEG-7 audio[18], [19] represents a standardized format to store audio description content. MPEG-7 defines a structured set of specific extensions to XML schema, which can store both low-level and high-level descriptors. The MPEG-7 serves as useful, implementation independent interface between the

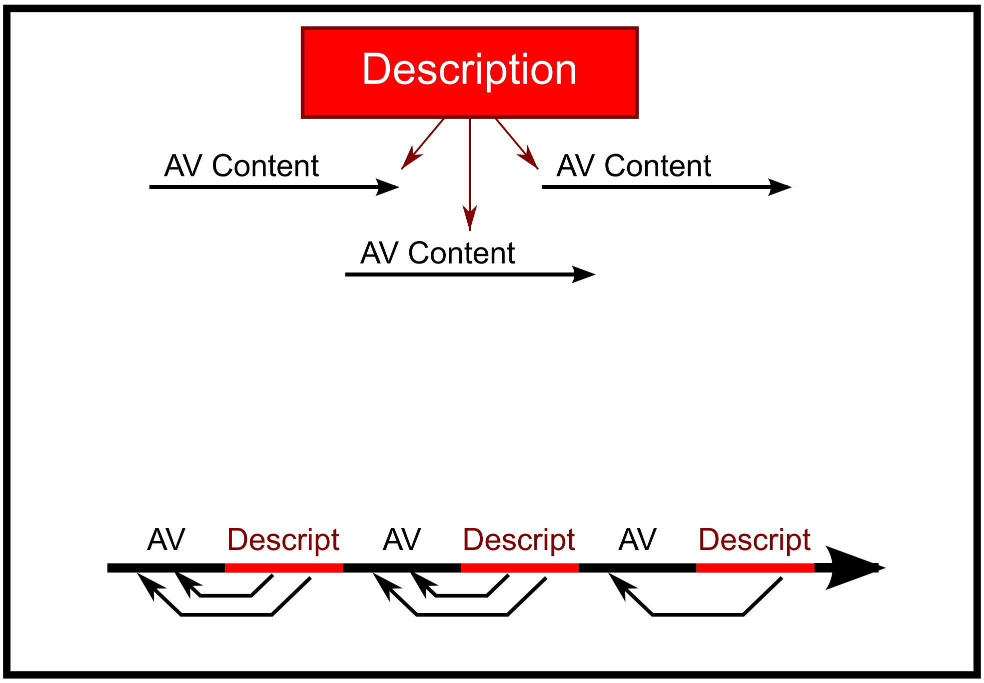

Figure 1.1: An MPEG-7 architecture requirement is that description must be separated from the audiovisual content. On the other hand, there must be a relation between the content and description. Thus the description is multiplexed with the content itself.[1]

audio describing tools and the software that utilizes the audio descriptors, including, of course, the methods of machine learning.

MPEG-7 additionally defines 17 low-level audio descriptors based on spectral and temporal audio features. Lampropoulos and Tsihrintzis [20] classify the descriptors in following groups:

1. Basic: Audio Power (AP), Audio Waveform(AWF).
   Temporally sampled scalar values for general use, applicable to all kinds of signals. The AP describes the temporally-smoothed instantaneous power of samples in the frame, in other words it is a temporal measurement of signal content as a function of time and offers a quick summary of a signal in conjunction with other basic spectral descriptors. The AWF describes audio waveform envelope (minimum and maximum), typically for display purposes.

2. Basic Spectral: Audio Spectrum Envelop (ASE), Audio Spectrum Centroid (ASC), Audio Spectrum Spread (ASS), Audio Spectrum Flatness (ASF).
   All share a common basis, all deriving from the short term audio signal spectrum (analysis of frequency over time). They are all based on the ASE Descriptor, which is a logarithmic-frequency spectrum. This descriptor provides a compact description of the signal spectral content and represents the similar approximation of logarithmic response of the human ear. The ASE descriptor is an indicator as to whether the spectral content of a signal is dominated by high or low frequencies. The ASC Descriptor could be considered as an approximation of perceptual sharpness of the signal. The ASS descriptor indicates whether the signal content, as it is represented by the power spectrum, is concentrated around its centroid or spread out over a wider range of the spectrum. This gives a measure which allows the distinction of noise-like sounds from tonal sounds. The ASF describes the flatness properties of the spectrum of an audio signal for each of a number of frequency bands.

3. Basic Signal Parameters: Audio Fundamental Frequency (AFF) and Audio Harmonicity (AH).
   The signal parameters constitute a simple parametric description of the audio signal. This group includes the computation of an estimate for the fundamental frequency (F0) of the audio signal. The AFF descriptor provides estimates of the fundamental frequency in segments in which the audio signal is assumed to be periodic. The AH represents the harmonicity of a signal, allowing distinction between sounds with a harmonic spectrum (e.g., musical tones or voiced speech e.g., vowels), sounds with an inharmonic spectrum (e.g., bell-like sounds) and sounds with a non-harmonic spectrum (e.g., noise, unvoiced speech).

4. Temporal Timbral: Log Attack Time (LAT), Temporal Centroid (TC).
   Timbre refers to features that allow one to distinguish two sounds that are equal in pitch, loudness and subjective duration. These descriptors are taking into account several perceptual dimensions at the same time in a complex way. Temporal Timbral descriptors describe the signal power function over time. The power function is estimated as a local mean square value of the signal amplitude value within a running window. The LAT descriptor characterizes the ”attack” of a sound, the time it takes for the signal to rise from silence to its maximum amplitude. This feature signifies the differ-

ence between a sudden and a smooth sound. The TC descriptor computes a timebased centroid as the time average over the energy envelope of the signal.

5. Timbral Spectral descriptors: Harmonic Spectral Centroid (HSC), Harmonic Spectral Deviation (HSD), Harmonic Spectral Spread (HSS), Harmonic Spectral Variation (HSV) and Spectral Centroid. These are spectral features extracted in a linear-frequency space. The HSC descriptor is defined as the average, over the signal duration, of the amplitude-weighted mean of the frequency of the bins (the harmonic peaks of the spectrum) in the linear power spectrum. It is has a high correlation with the perceptual feature of "sharpness" of a sound. The HSD descriptor measures the spectral deviation of the harmonic peaks from the global envelope. The HSS descriptor measures the amplitude-weighted standard deviation (Root Mean Square) of the harmonic peaks of the spectrum, normalized by the HSC. The HSV descriptor is the normalized correlation between the amplitude of the harmonic peaks between two subsequent time-slices of the signal.

6. Spectral Basis, which consists of Audio Spectrum Basis (ASB) and Audio Spectrum Projection (ASP).

1.3.2 Music Features

Music features are combination of MPEG-7 descriptors: 1.3.1 and other features. These features are extracted to represent five perceptual dimensions of music listening: energy, rhythm, temporal, spectrum and melody. The following are groups of features(without MPEG-7) [21]:

1. Energy features: Specific loudness sensation coefficients (SONE), Total loudness (TL).
   The resulting power spectrum, which reflects human loudness sensation better than AP, is called sonogram. SONE are the coefficients computed from sonogram, which consists of up 24 Bark (1.2) critical bands (the actual number of critical bands depends on the sampling frequency of the audio signal). TL is computed as an aggregation of SONE based on Steven's method[22] which takes the sum of the largest SONE coefficient and 0.15 ratio of the sum of the remainder coefficients.

2. Temporal features: Zero crossing rate (ZCR).
   ZCR a measure of the signal noisiness, is computed by taking the mean

and standard deviation of the number signal values that cross the zero axis in each time window(i.e., sign changes).

3. Spectrum features: Mel-frequency cepstral (MFCC) coefficients, Spectral Contrast.
MFCC are commonly used timbre feature, the coefficients of the discrete cosine transform of each short-term log power spectrum expressed on a nonlinear perception-related Mel-frequency scale. It represents the formant peaks of the spectrum.

   Octave-based spectral contrast to capture the relative energy distribution of the harmonic components in the spectrum . The feature considers the spectral peak, spectral valley, and their dynamics in each subband and roughly reflects the relative distribution of the harmonic and non-harmonic components in the spectrum

4. Harmony feature: Chroma
Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave

1.4 Classifiers and Classes

The machine learning techniques in this thesis solve a problem formally known as classification. In [23], it is described as follows:

> Formally, a classifier is a mapping of some feature space $\mathscr{X}$ to some collection of classes $c_1, \ldots, c_m$,
>
> $$\phi : \mathscr{X} \to C = \{c_1, \ldots, c_m\}. \tag{1.1}$$
>
> The collection C is sometimes called classification of $\mathscr{X}$, though more frequently, the term classification denotes the process of constructing a classifier ϕ and subsequently using it to predict the class of unseen inputs $x \in \mathscr{X}$. Several important aspects of that process will be discussed in the remaining sections of this chapter. Here, on the other hand, we will have a closer look at the domain and value set of the mapping (1.1).
>
> 1. The *feature space* $\mathscr{X}$ is the space from which the combinations $x = ([x]_1, \ldots, [x]_n)$ of values of input features are taken. Hence, it is the Cartesian product $V_1 \times \cdots \times V_n$ of sets $V_1, \ldots, V_n$ of feasible values of the individual features. However, it is important that not every combination $([x]_1, \ldots, [x]_n)$ from the Cartesian product

of sets of feasible values is a feasible combination: imagine a recommender system and the combination of `Client Age = 10` and `Client Marital Status = divorced`. Hence, the domain of the classifier ϕ is in general not the whole $V_1 \times \cdots \times V_n$, but only some subset of it,

$$\mathrm{Dom}\phi = \mathscr{X} \subset V_1 \times \cdots \times V_n. \tag{1.2}$$

The *features* $[x]_1, \ldots, [x]_n$ are alternatively called also *attributes* or *variables*, and their number can be quite high: several thousands are not an exception. From the point of view of data types, they can be very diverse, e.g.:

- *Continuous* data, such as real numbers, sound energy of speech or music, intensity of light.
- *Ordinal* data, such as various preferences, lexicographically ordered parts of text.
- *Categorical* data, aka *nominal* data, such as sex, place of residence, or colour, with a finite set V of feasible values. The elements of V are called categories.
- *Binary* data, such as sex, are categorical data for which the cardinality $|V|$ of the set V fulfils $|V| = 2$. They are, of course, a specific kind of categorical data, but at the same time, any categorical data can be always represented by a vector of binary data, usually of the binary data with the value set $\{0, 1\}$. Indeed, if the $|V|$ elements of V are enumerated as $v_1, \ldots, v_{|V|}$, then the element v_j can be represented by a vector $b_j \in \{0, 1\}|V|$ such that

$$[b_j]_j = 1, [b_j]_k = 0 \text{ for } k \neq j. \tag{1.3}$$

2. The collection of *classes* $C = \{c_1, \ldots, c_m\}$ is always finite. Most common is the case $m = 2$, called *binary classification*, e.g., spam and ham, products to be recommended and those not to be recommended, malware and harmless software, network intrusion and normal traffic. For binary classification, a different notation is frequently employed, e.g., $C = \{c_+, c_-\}, C = \{1, 0\}, C = \{1, -1\}$, the first of the involved cases being called *positive*, the second *negative*. The case $m = 3$ is sometimes obtained from binary classification through introducing an additional class for those cases causing difficulties to the classifier. The interpretation of such a class then means" to some degree positive, to some certain degree negative".

1.5 Measures of Classifier Performance

M. Holeňa in [23] additionally introduces the metrics used to evaluate and compare the performance of classifiers:

> When solving a particular classification task, we typically have a large number of classifiers available. What helps to choose the most suitable one is on the one hand understanding their principles and underlying assumptions, on the other hand comparing different of them on the relevant data. Each such comparison has two ingredients:
>
> **(i)** A set, or more generally a sequence $x_1, \dots, x_q$ of independent inputs from the feature space such that for each $x_k, k = 1, \dots, q$, we know the correct class c_k. For the comparison based on the pairs $(x_1, c_1), \dots, (x_q, c_q)$ not to be biased, they must be selected independently of those used as the classifier was constructed. If $(x_1, c_1), \dots, (x_q, c_q)$ have been selected in this way, then they are usually called *test data*.
>
> **(ii)** A function evaluating the performance of the classifier on $(x_1, c_1), \dots, (x_q, c_q)$. The value of that function has usually the meaning of some error that the classifier ϕ makes when classifying $x_1, \dots, x_q$. Therefore, a generic function of that kind will be in the following denoted as ER_ϕ.
>
> The function ER_ϕ depends both on the test data $(x_1, c_1), \dots, (x_q, c_q)$ and on the classes $\phi(x_1), \dots, \phi(x_q)$ predicted for $x_1, \dots, x_q$ by ϕ. Thus if we restrict attention to crisp classifiers (1.1), then in general,
>
> $$\mathrm{ER}_\phi : \mathscr{X} \times C \times C \to \mathbb{R}. \tag{1.4}$$
>
> Frequently, ER_ϕ depends on $x_1, \dots, x_q$ only through the predictions $\phi(x_1), \dots, \phi(x_q)$, hence
>
> $$\mathrm{ER}_\phi : C \times C \to \mathbb{R}. \tag{1.5}$$
>
> In such a case, ER_ϕ is completely determined by the counts of data with the correct class c_i and classified to the class c_j,
>
> $$q_i = |\{k | 1 \leq k \leq q, c_k = c_i, \phi(x_k) = c_j\}|, i, j = 1, \dots, m \tag{1.6}$$
>
> Together with the overall count of test data with the correct class c_i, and the overall count of test data classified to c_j,
>
> $$q_{i\cdot} = \sum_{j=1}^{m} q_{i,j}, \text{ respectively } q_{\cdot j} = \sum_{i=1}^{m} q_{i,j}, i, j = 1, \dots, m \tag{1.7}$$

they form the following matrix, called *confusion matrix* of the classifier ϕ:

$$\begin{array}{c|ccc} q & q_{\cdot 1} & \dots & q_{\cdot m} \\ \hline q_{1\cdot} & q_{1,1} & \dots & q_{1,m} \\ \dots & \dots & & \dots \\ q_{m\cdot} & q_{m,1} & \dots & q_{m,m} \end{array} \tag{1.8}$$

The most commonly encountered function of the kind1.5 is *classification error* – the proportion of test data for which $\phi(x_k) \neq c_k$:

$$\mathrm{ER}_\phi = \mathrm{ER}_{CE} = \frac{1}{m} \sum_{i \neq j} q_{i,j}. \tag{1.9}$$

The complementary proportion of test data for which $\phi(x_k) = c_k$ is called *accuracy*, or frequently *predictive accuracy*, to emphasize that it means the prediction of correct class for the unseen test data,

$$\mathrm{AC} = \frac{1}{m} \sum_{i=1}^{m} q_{i,i} = 1 - \mathrm{ER}_{CE}. \tag{1.10}$$

Notice that according to (1.9) and (1.10), all erroneous classifications $\phi(x_k) \neq c_k$ contribute to ER_{CE} equally. This corresponds to an assumption that all kinds of erroneous classifications are equally undesirable. Therefore, a *weighted error* (or cost-weighted error) is used as a more realistic counterpart of (1.9)

$$\mathrm{ER}_\phi = \mathrm{ER}_W = \frac{1}{m} \sum_{i=1}^{m} \sum_{j \neq i} w_{i,j} q_{i,j}, \tag{1.11}$$

where $w_{i,j}, i, j = 1, \dots, m$,denotes the weight or cost of the misclassification $\phi(x_k) = c_j$ if the correct class is c_i. Formally, also a cost of correct classification can be introduced, $w_{i,i}, i = 1, \dots, m$, normally set to $w_{i,i} = 0$, which simplifies (1.11) to

$$\mathrm{ER}_\phi = \mathrm{ER}_W = \frac{1}{m} \sum_{i,j=1}^{m} w_{i,j} q_{i,j}. \tag{1.12}$$

The traditional classification error then corresponds to the classification cost

$$w_{i,j} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases} \tag{1.13}$$

Frequently, the costs $w_{i,j}$ are scaled so that $\sum_{i,j=1}^{m} w_{i,j} = 1$. This is always possible through dividing them by the original $\sum_{i,j=1}^{m} w_{i,j}$. It turns the costs to a probability distribution on the pairs $(i,j)_{i,j=1}^{m}$ and the cost-weighted error (1.12) to the mean value of classification error with respect to that distribution. For the traditional classification error (1.9), these scaled costs are $w_{i,j} = \frac{1}{m(m-1)}, i \neq j$.

1.5.1 Performance Measures in Binary Classification

In the case of a binary classifier $\phi : \mathscr{X} \rightarrow \{c_+, c_-\}$,there are only 4 possible values $q_{i,j}$,which have got their specific names, introduced below in Table 1.1. Frequently, they are used as rates with respect to the overall number $q_{+\cdot}$ assigned to the class c_+ and the overall number $q_{-\cdot}$ assigned to the class c_-, as is also explained in Table 1.1. By means of the values in this table, classification error (1.9) can be rewritten as

$$\mathrm{ER}_{CE} = \frac{1}{q}(\mathrm{FP} + \mathrm{FN}), \tag{1.14}$$

accuracy (1.10) as

$$\mathrm{AC} = \frac{1}{q}(\mathrm{TP} + \mathrm{TN}), \tag{1.15}$$

and cost-weighted error(1.12), using a notation analogous to wi, j for a classification into the classes c_+,c_-, as

$$\mathrm{ER}_{W} = \frac{1}{q}(w_{++}\mathrm{TP} + w_{+-}\mathrm{FN} + w_{-+}\mathrm{FP} + w_{--}\mathrm{TN}). \tag{1.16}$$

Apart from (1.14)–(1.16), also the true positive rate TPr, false positive rate FPr, true negative rate TNr and the additional measures precision and F-measure are often used as performance measures in binary classification. *Precision* PR is defined

$$\mathrm{PR} = \frac{\mathrm{TP}}{q_{\cdot+}}, \tag{1.17}$$

the definition of the F-measure FM is

$$\mathrm{FM} = 2\frac{\mathrm{PR}{\cdot}\mathrm{TPr}}{\mathrm{PR}{+}\mathrm{TPr}}. \tag{1.18}$$

Due to the ubiquity of binary classification, several of its performance measures are known also under alternative names. The most important among such synonyms are as follows:

Table 1.1: Confusion matrix in binary classification

$q = q_{\cdot+} + q_{\cdot-}$ $= q_{+\cdot} + q_{-\cdot}$		Classified as $c_+ : q_{\cdot+}$	$c_- : q_{\cdot-}$	Rate (r)	
Correct	$c_+ : q_{+\cdot}$	true positive (TP)	false negative (FN)	TPr=$\frac{\text{TP}}{q_{+\cdot}}$	FNr=$\frac{\text{FN}}{q_{+\cdot}}$
class	$c_- : q_{-\cdot}$	false positive (FP)	true negative (TN)	FPr=$\frac{\text{FP}}{q_{-\cdot}}$	TNr=$\frac{\text{TN}}{q_{-\cdot}}$

- *predictive value* is a synonym for precision,
- *sensitivity* and *recall* are synonyms for true positive rate,
- *specificity* is a synonym for true negative rate.

In binary classification, classifier performance is very often characterized not by a single performance measure, but by two such measures simultaneously. Most common are the pairs of measures (FPr,TPr), (AC,PR) a (PR,TPr). Notice that an ideal classifier, i.e., one for which true positive rate is 1 and false positive rate is 0, has the following values of those three pairs of measures:

$$(\text{FPr,TPr}) = (0, 1), (\text{AC,PR}) = (1, 1), (\text{PR,TPr}) = (1, 1). \tag{1.19}$$

A pair of performance measures is particularly useful in the following situations:

(i) The performance of a classifier has been measured with different test data, typically with different subsequences of the sequence $(x_1, c_1), \ldots, (x_q, c_q)$.

(ii) The performance has been measured not for a single classifier, but for a set of classifiers, typically classifiers of the same kind, differing through the values of one or several parameters.

In both situations, the resulting pairs form a set in the 2-dimensional space, which can be connected with a curve according to increasing values of one of the two involved measures. For the pair of measures (FPr,TPr), such curves are called *receiver operating characteristics* (ROC) because they were first proposed for classification tasks in radar detection. If the ROC curve is constructed in the situation (i), then it provides an additional performance measure of the considered classifier. The area under the ROC curve, i.e. the area delimited from above by the curve, from below by the value TPr=0 and from the left and right by the values

FPr=0 and FPr=1,has the size AUC= $\int_0^1$ TPr dFPR. Because the highest possible value of TPr is 1, AUC is delimited by

$$\mathrm{AUC} = \int_0^1 \mathrm{TPr}\ \mathrm{dFPR} \leq \int_0^1 1\mathrm{dFPr} = 1 \tag{1.20}$$

This performance measure summarizes the pairs of measures obtained for several sequences of test data (those used to construct the ROC curve) into one value.

1.6 Employed Classification Methods

We have elaborated our approach to sentiment analysis from utterances for six classification methods: k nearest neighbors, support vector machines, multi-layer perceptrons, classification trees, random forests [24] and long short-term memory (LSTM) networks [25, 26, 27].

1.6.1 k Nearest Neighbours (kNN)

A very traditional way of classifying a new feature vector $x \in \mathcal{X}$ if a sequence of training data $(x_1, c_1), \ldots, (x_p, c_p)$ is available is the nearest neighbour method: take the x_j that is the closest to x among $x_1, \ldots, x_p$, and assign to x the class assigned to x_j, i.e., c_j.

A straightforward generalization of the nearest neighbour method is to take among $x_1, \ldots, x_p$ not one, but k feature vectors $x_{j_j}, \ldots, x_{j_k}$ closest to x. Then x is assigned the class $c \in C$ fulfilling

$$|\{i, 1 \leq i \leq k | c_{j_i} = c\}| = max_{c' \in C} |\{i, 1 \leq i \leq k | c_{j_i} = c'\}|. \tag{1.21}$$

This method is called, expectedly, k nearest neighbours, or k-NN for short.

1.6.2 Support Vector Machines (SVM)

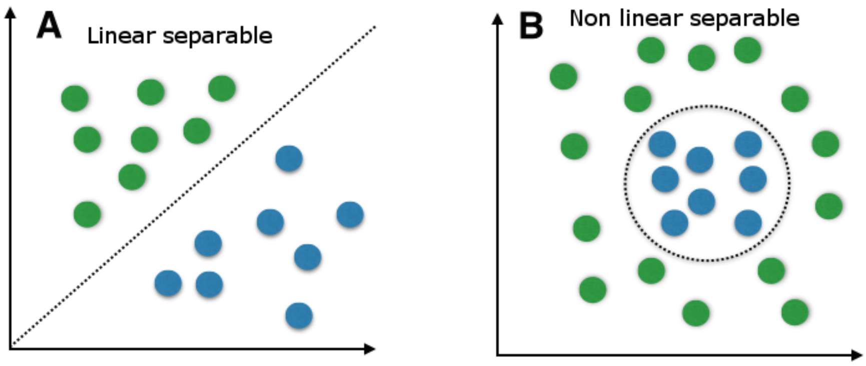

Figure 1.2: In figure A, the target labels can be separated with a line. This is called linear separability and can be done not only by SVM, but by any classifier whose output is a linear combination of input features. The figure B shows data which cannot be linearly separated, at least not in 2D space. As a linear classifier, SVM achieves the non-linear classification by adding another dimension to the data and separating them in the higher dimension. In the figure B, the data would be moved to 3D space and linearly separated by a plane.[2]

Support vector machines are classifiers into two classes. This method attempts to derive from the training data $(x_1, c_1), \ldots, (x_p, c_p)$ the best possible generalization to unseen feature vectors.

If both classes, more precisely their intersections with the set $\{x_1, \ldots, x_p\}$ of training inputs, are in the space of feature vectors linearly separable, the method constructs two parallel hyperplanes $H_+ = \{x \in \mathbb{R}^n | x^\top w + b_+ = 0\}, H_- = \{x \in \mathbb{R}^n | x^\top w + b_- = 0\}$ such that the training data fulfil

$$c_k = \begin{cases} 1 & \text{if } x^\top w + b_+ \geq 0, \\ \text{-1} & \text{if } x^\top w + b_- \leq 0, \end{cases} k = 1, \ldots, p, \tag{1.22}$$

$$H_+ \cap \{x_1, \ldots, x_p\} \neq \emptyset, H_- \cap \{x_1, \ldots, x_p\} \neq \emptyset. \tag{1.23}$$

The hyperplanes H_+ and H_- alle called support hyperplanes. Their common normal vector w and intercepts b_+, b_- are obtained through solving the following constrained optimization task:

Maximize with respect to w, b_+, b_- the distance

$$d(H_+, H_-) = \frac{b_+ - b_-}{\|w\|} \tag{1.24}$$

on condition that the p inequalities (1.22) hold.

The distance (1.24) is commonly called margin. The solution to this optimization task coincides with the $(w^*, b^*_+, b^*_-, \alpha^*_1, \ldots, \alpha^*_p)$ of the Lagrange function

$$L(w, b_+, b_-, \alpha_1, \ldots, \alpha_p) = \|w\|^2 + \sum_{k=1}^{p} \alpha_k(\frac{b_+ - b_-}{2} - c_k x_k^\top w) \tag{1.25}$$

where $\alpha_1, \ldots, \alpha_p \geq 0$ are Lagrange coefficients of the optimization task. Once the saddle point $(w^*, b^*_+, b^*_-, \alpha^*_1, \ldots, \alpha^*_p)$ is found, the classifier is defined by

$$\phi(x) = \begin{cases} 1 & \text{if } \sum_{x_k \in \mathcal{S}} \alpha^*_k c_k x^\top x_k + b^* \geq 0, \\ -1 & \text{if } \sum_{x_k \in \mathcal{S}} \alpha^*_k c_k x^\top x_k + b^* < 0, \end{cases} \tag{1.26}$$

where $b^* = \frac{1}{2}(b^*_+ + b^*_-)$ and

$$\mathcal{S} = \{x_k | \alpha^*_k > 0\}. \tag{1.27}$$

Due to the Karush-Kuhn-Tucker (KKT) conditions,

$$\alpha^*_k(\frac{b^*_+ - b^*_-}{2} - c_k x_k^\top w^*) = 0, k = 1, \ldots, p, \tag{1.28}$$

all feature vectors from the set $\mathcal{S}$ lie on some of the suport hyperplanes (1.23). Therefore, they are called support vectors. This name together with the observation that they completely determine the classifier defined in (1.26) explains why such a classifier is called support vector machine.

If the intersections of both classes with the training inputs are not linearly separable, a SVM is constructed similarly, but instead of the set of possible fature vectors, now the set of functions

$$\kappa(\cdot, x) \text{ for all possible feature vectors } x \tag{1.29}$$

is considered, where κ is a kernel, i.e., a mapping on pairs of feature vectors that is symmetric and such that for any $k \in \mathbb{N}$ and any sequence of different feature vectors $x_1, \ldots, x_k$, the matrix

$$G_\kappa(x_1, \ldots, x_k) = \begin{pmatrix} \kappa(x_1, x_1) & \ldots & \kappa(x_1, x_k) \\ \ldots & \ldots & \ldots \\ \kappa(x_k, x_1) & \ldots & \kappa(x_k, x_k) \end{pmatrix}, \tag{1.30}$$

which is called the Gramm matrix of $x_1, \ldots, x_k$, is positive semidefinite, i.e.,

$$(\forall y \in \mathbb{R}^k) \; y^\top G_\kappa(x_1, \ldots, x_k) y \geq 0. \tag{1.31}$$

The most commonly used kinds of kernels are the Gaussian kernel with a parameter $\varsigma > 0$,

$$(\forall x, x' \in \mathbb{R}^{n'}) \; \kappa(x, x') = \exp\left(-\frac{1}{\varsigma}\|x - x'\|^2\right), \tag{1.32}$$

and polynomial kernel with parameters $d \in \mathbb{N}$ and $c \geq 0$,

$$(\forall x, x' \in \mathbb{R}^{n'})\ \kappa(x, x') = (x^\top x' + c)^d. \tag{1.33}$$

It is known [28] that, due to the properties of kernels, if the joint distribution of a sequence of different feature vectors $x_1, \dots, x_k$ is continuous, then almost surely any proper subset of the set of functions $\{\kappa(\cdot, x_1), \dots, \kappa(\cdot, x_k)\}$ is in the space of all functions (1.29) linearly separable from its complement.

However, the featre vectors x and x_k can't be simply replaced by the corresponding functions $\kappa(\cdot, x)$ and $\kappa(\cdot, x_k)$ in the definition (1.26) of a SVM classifier because a transpose $x^\top$ exists for a finite-dimensional vector, but not a for an infinite-dimensional function. Fortunately, the transpose occurs in (1.26) only as a part of the scalar product $x^\top x_k$. And a scalar product can be defined also on the space of all functions (1.29). Namely, the properties of a scalar product has the function that to the pair of functions $(\kappa(\cdot, x), \kappa(\cdot, x')$ assigns the value $\kappa(x, x')$. Using this scalar product in (1.26), we obtain the following definition of a SVM classifier for linearly non-separable classes:

$$\phi(x) = \begin{cases} 1 & \text{if } \sum_{x_k \in \mathcal{S}} \alpha_k^* c_k \kappa(x, x_k) + b \geq 0, \\ -1 & \text{if } \sum_{x_k \in \mathcal{S}} \alpha_k^* c_k \kappa(x, x_k) + b \geq 0. \end{cases} \tag{1.34}$$

1.6.3 Multilayer Perceptrons (MLP)

A multilayer percptron is a mapping ϕ of feature vectors to classes with which a directed graph $G_\phi = (\mathcal{V}, \mathcal{E})$ is associated. Due to the inspiration from biological neural networks, the vertices of G_ϕ are called *neurons* and its edges are called *connections*. In addition, G_ϕ is required to have a layered structure, which means that the set $\mathcal{V}$ of neurons can be decomposed into $L+1$ mutually disjoint layers, $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1 \cup \dots \cup \mathcal{V}_L, L \geq 2$, such that

$$(\forall (u, v) \in \mathcal{E})\ u \in \mathcal{V}_i, i = 0, \dots, L-1\ \&\ v \notin \mathcal{V}_i \Rightarrow v \in \mathcal{V}_{i+1}. \tag{1.35}$$

The layer $\mathcal{I} = \mathcal{V}_0$ is called input layer of the MLP, the layer $\mathcal{O} = \mathcal{V}_L$ its output layer and the layers $\mathcal{H}_1 = \mathcal{V}_1, \dots, \mathcal{H}_{L-1} = \mathcal{V}_{L-1}$ its hidden layers.

The purpose of the graph G_ϕ associated with the mapping ϕ is to define a decomposition of ϕ into simple mappings assigned to hidden and output neurons and to connections between neurons (input neurons normally only accept the components of the input, and no mappings are assigned to them). Inspired by biological terminology, mappings assigned to neurons are called *somatic*, those assigned to connections are called *synaptic*.

To each connection $(u, v) \in \mathcal{E}$, the multiplication by a weight $w_{(u,v)}$ is assigne as a synaptic mapping:

$$(\forall \xi \in \mathbb{R})\ f_{(u,v)}(\xi) = w_{(u,v)} \xi. \tag{1.36}$$

To each hidden neuron $v \in \mathcal{H}_i$, the following somatic mapping is assigned:

$$(\forall \xi \in \mathbb{R}^{|\,in(v)|})\ f_v(\xi) = \varphi(\sum_{u \in in(v)} [\xi]_u + b_v), \tag{1.37}$$

where $[\xi]_u$ for $u \in in(v)$ denotes the component of ξ that is the output of the synaptic mapping $f_{u,v}$ assigned to the connection (u,v), $in(v) = \{u \in \mathcal{V} | (u,v) \in \mathcal{E}\}$ is the input set of v, and $\varphi : \mathbb{R} \to \mathbb{R}$ is called activation function. Though the activation functions, in applications typically sigmoidal functions are used to this end, i.e., functions that are non-decreasing, piecewise continuous, and such that

$$-\infty < \lim_{t \to -\infty} \varphi(t) < \lim_{t \to \infty} \varphi(t) < \infty. \tag{1.38}$$

The activation functions most frequently encountered in MLPs are:

- the logistic function,

$$(\forall t \in \mathbb{R})\ \varphi(t) = \frac{1}{1 + \mathrm{e}^{-t}}; \tag{1.39}$$

- the hyperbolic tangent,

$$\varphi(t) = \tanh t = \frac{\mathrm{e}^{t} - \mathrm{e}^{-t}}{\mathrm{e}^{t} + \mathrm{e}^{-t}}. \tag{1.40}$$

To an output neuron $v \in \mathcal{O}$, also a somatic mapping of the kind (1.37) with the activation functions (1.39) or (1.40) can be assigned. If it is the case, then the class c predicted for a feature vector x is obtained as $c = \arg\max_i(\phi(x))_i$, where $(\phi(x))_i$ denotes the i-the component of $\phi(x)$. Alternatively the activation function assigned to an output neuron can be the step function, aka Heaviside function

$$\varphi(t) = \begin{cases} 0 & \text{if } t < 0, \\ 1 & \text{if } t \geq 0. \end{cases} \tag{1.41}$$

In that case, the value $(\phi(x))_c$ already directly indicates whether x belongs to the class c.

1.6.4 Classification Trees (CT)

A classifier $\phi : \mathcal{X} \to C = \{c_1, \ldots, c_m\}$ is called binary classification tree, if there is a binary tree $T_\phi = (V_\phi, E_\phi)$ with vertices V_ϕ and edges E_ϕ such that:

(i) $V_\phi = \{v_1, \ldots, v_L, \ldots, v_{2L-1}\}$, where $L \geq 2$, v_0 is the root of T_ϕ, $v_1, \ldots, v_{L-1}$ are its forks and $v_L, \ldots, v_{2L-1}$ are its leaves.

(ii) If the children of a fork $v \in \{v_1, \ldots, v_{L-1}\}$ are $v^L \in V_\phi$ (left child) and $v^R \in V_\phi$ (right child) and if $v = v_i, v^L = v_j, v^R = v_k$, then $i < j < k$.

(iii) To each fork $v \in \{v_1, \dots, v_{L-1}\}$, a predicate φ_v of some formal logic is assigned, evaluated on features of the input vectors $x \in \mathcal{X}$.
(iv) To each leaf $v \in \{v_L, \dots, v_{2L-1}\}$, a class $c_v \in C$ is assigned.
(v) For each input $x \in \mathcal{X}$, the predicate φ_{v_1} assigned to the root is evaluated.
(vi) If for a fork $v \in \{v_1, \dots, v_{L-1}\}$, the predicate φ_v evaluates true, then $\phi(x) = c_{v^L}$ in case v^L is already a leaf, and the predicate φ_{v^L} is evaluated in case v^L is still a fork.
(vii) If for a fork $v \in \{v_1, \dots, v_{L-1}\}$, the predicate φ_v evaluates false, then $\phi(x) = c_{v^R}$ in case v^R is already a leaf, and the predicate φ_{v^R} is evaluated in case v^R is still a fork.

1.6.5 Random Forests (RF)

Random Forests are ensembles of classifiers in which the individual members are classification trees. They are constructed by bagging, i.e., bootstrap aggregation of individual trees, which consists in training each member of the ensemble with another set of training data, sampled randomly with replacement from the original training pairs $(x_1, c_1), \dots, (x_p, c_p)$. Typical sizes of random forests encountered in applications are dozens to thousands trees. Subsequently, when new subjects are input to the forest, each tree classifies them separately, according to the leaves at which they end, and the final classification by the forest is obtained by means of an aggregation function. The usual aggregation function of random forests is majority voting, or some of its fuzzy generalizations.

According to which kind of randomness is involved in the costruction of the ensemble, two broad groups of random forests can be differentiated:

1. *Random forests grown in the full input space.* Each tree is trained using all considered input features. Consequently, any feature has to be taken into account when looking for the split condition assigned to an inner node of the tree. However, features actually occurring in the split conditions can be different from tree to tree, as a consequence of the fact that each tree is trained with another set of training data. For the same reason, even if a particular feature occurs in split conditions of two different trees, those conditions can be assigned to nodes at different levels of the tree.

   A great advantage of this kind of random forests is that each tree is trained using all the information available in its set of training data. Its main disadvantage is high computational complexity. In addition, if several or even only one variable are very noisy, that noise gets nonetheless incorporated into all trees in the forest. Because of those disadvantages, random forests are grown in the complete input space primarily if its dimension is not high and no input feature is substantially noisier than the remaining ones.

2. *Random forests grown in subspaces of the input space.* Each tree is trained using only a randomly chosen fraction of features, typically a small one. This means that a tree t is actually trained with projections of the training data into a low-dimensional space spanned by some randomly selected dimensions $i_{t,1} \leq \cdots \leq i_{t,d_t} \in \{1, \ldots, d\}$, where d is the dimension of the input space, and d_t is typically much smaller than d. Using only a subset of features not only makes forest training much faster, but also allows to eliminate noise originating from only several features. The price paid for both these advantages is that training makes use of only a part of the information available in the training data.

1.6.6 Long Short-Term Memory (LSTM)

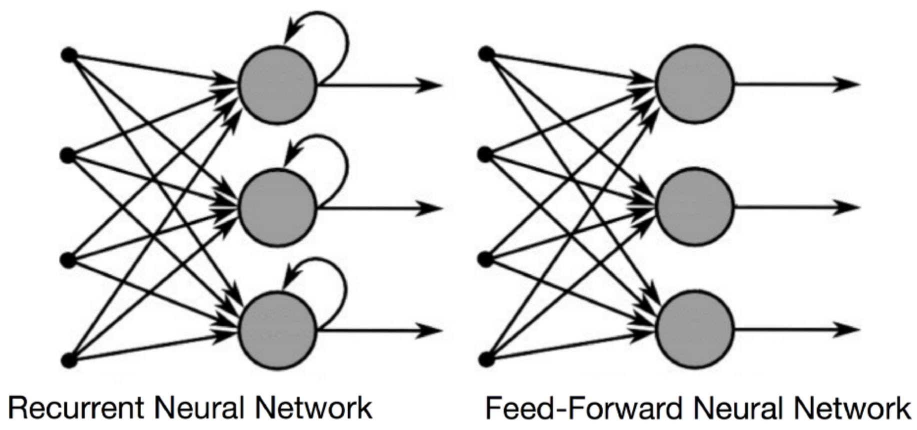

Figure 1.3: The difference in the information flow between a RNN and a Feed-Forward Neural Network. [3]

An LSTM network is used for classification of sequences of feature vectors, or equivalently, multidimensional time series with discrete time. Alternatively, it can be also employed to obtain sequences of such classifications, i.e., in situations when the neural network input is a sequence of feature vectors and its output is a a sequence of classes. Differently to most of other commonly encountered kinds of artificial neural networks, an LSTM layer connects not simple neurons, but units with their own inner structure. Several variants of an LSTM have been proposed (e.g., [25, 26]), all of them include at least the following four kinds of units described below. Each of them has certain properties of usual ANN neurons, in particular, the values assigned to them depend, apart from a bias, on values assigned to the unit input at the same time step and on values assigned to the unit output at the previous time step. Hence, an LSTM network layers is a recurrent network.

(i) *Memory cells* can store values, aka cell states, for an arbitray time. They have no activation function, thus their output is actually a biased linear combination of unit inputs and of the values from the previous time step coming through recurrent connections.

(ii) *Input gate* controls the extent to which values from the previous unit or from the preceding layer influence the value stored in the memory cell. It has a sigmoidal activation function, which is applied to a biased linear combination of unit inputs and of values from the previous time step, though the bias and synaptic weights of the input and recurrent connections are specific and in general different from the bias and synaptic weights of the memory cell.

(iii) *Forget gate* controls the extent to which the memory cell state is supressed. It again has a sigmoidal activation function, which is applied to a specific biased linear combination of unit inputs and of values from the previous time step.

(iv) *Output gate* controls the extent to which the memory cell state influences the unit output. Also this gate has a sigmoidal activation function, which is applied to a specific biased linear combination of unit inputs and of values from the previous time step, and subsequently composed either directly with the cell state or with its sigmoidal transformation, using another sigmoid than is used by the gates.

1.7 Verification methods

1.7.1 Cross validation

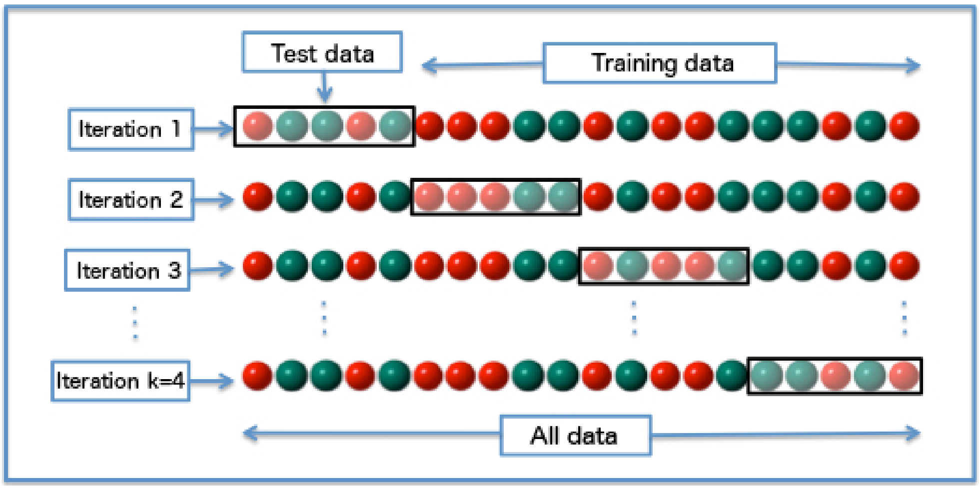

Figure 1.4: Diagram of k-fold cross-validation with $k = 4$. [4]

Cross-validation is a systematic use of available data that allows to measure the performance of a statistical method. Its core idea is to iteratively split all data into two disjunct sets, the *training set* and the *validation set*. For each iteration, model parameters are optimized to fit the training set. This model is then used to evaluate the data from validation set, where the difference between predicted and actual values represent a measure of fit for the iteration. The combination of measures of fit over all iterations provide the estimated performance of the statistical method on unknown data.

In k-fold cross-validation, the data is shuffled and partitioned into k sets of equal size. The cross-validation then runs in k iterations, where each of the k sets is used exactly once as a validation set, while the remaining $k - 1$ sets serve as training data. Each iteration produces one measure of fit. These are then combined to produce a single performance estimation. The bias of k-fold cross-validation depends on a careful choice of k.

Stratified k-fold cross-validation is a modified version typically used for classification problems. In this version, the partitioning of data is done so that the distribution of all classes is roughly the same in each of the k sets.

In repeated random cross-validation, also called Monte Carlo cross-validation, both the size of validation set and its content are chosen randomly. [29]

1.7.2 Friedman test

Friedman test compares the performance of k classifiers on N datasets and determines, whether there are statistically significant differences in the performance of the classifiers. Demsar [5] defines:

> The Friedman test [30] is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, etc. In case of ties, average ranks are assigned.
>
> Let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic
>
> $$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{1.42}$$
>
> is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$).

1.7.3 Holm correction

The Bonferroni-Holm correction [7] is used when multiple hypotheses are tested. Bonferroni correction re-calculates the rejection criteria of all hypotheses according to desired FWER (which is a level of significance α shared by all hypotheses). Holm correction uses a different re-calculation method, which additionally reduces the Type II error.

The method is defined according to [31]:

> - Let $H_1, \dots, H_m$ be a family of m null hypotheses and $P_1, \dots, P_m$ the corresponding p-values.
> - Start by ordering the p-values (from lowest to highest) $P_{(1)} \dots P_{(m)}$ and let the associated hypotheses be $H_{(1)} \dots H_{(m)}$
> - For a given significance level α, let k be the minimal index such that $P_{(k)} > \frac{\alpha}{m+1-k}$
> - Reject the null hypotheses $H_{(1)} \dots H_{(k-1)}$ and do not reject $H_{(k)} \dots H_{(m)}$

- If $k = 1$ then do not reject any of the null hypotheses and if no such k exist then reject all of the null hypotheses.

Chapter 2

Approach

2.1 Available tools

2.1.1 Tools for Working with MPEG-7 Descriptors

We utilized the Sound Description Toolbox [11] and MPEG-7 Audio Analyzer - Low Level Descriptors Extractor [32] for our experiments. Both of them extract a number of MPEG-7 standard descriptors, both scalar ones and a time series. In addition, the SDT also calculates perceptual features such as Mel Frequency Cepstral Coefficients, Specific Loudness and Sensation Coefficients. From these descriptors SDT calculates means, covariances, means of first-order differences and covariances of first order differences. The Total number of features provided by this toolbox is 187.

2.1.2 Tools for Working with music features

LibROSA[33] is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. Outputs are time series.

2.1.3 np2mat

np2mat[34] is function for convert python (Numpy) ndarray to Matlab matrix.

2.2 Workflow

SDT and cross-validation (Fig: 2.1)

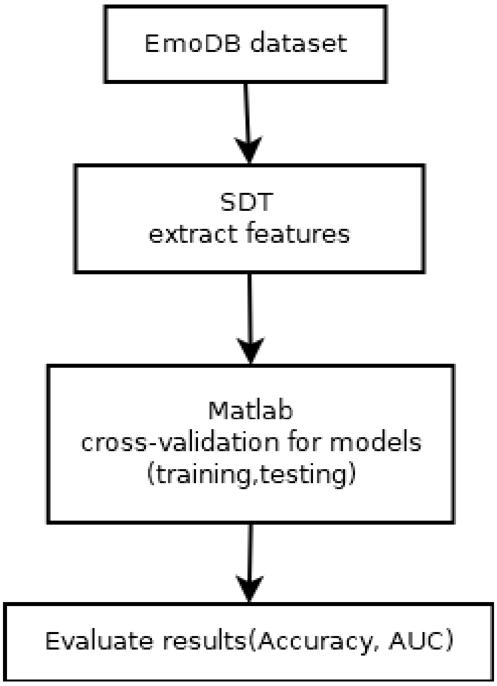

Figure 2.1: An overview of the sentiment analysis process.

Algorithm 1 SDT and cross validation algorithm for evaluating models on emotions

```
procedure SDTAndCrossValidation
    extract scalar values from audio files from EmoDB with SDT.
    load scalar values items and their classification to Matlab.
    for each model (kNN, SVM, MLP, DT, RF) M_i do
        for each cross-validation fold F_i do
            train M_i on training data not included in F_i
            calculate accuracy from M_i on testing data F_i
            calculate AUC for each emotion on testing data from F_i
        calculate average accuracy and AUC for M_i.
    return accuracy and AUC results.
```

SDT and Friedman test (Fig:2.2)

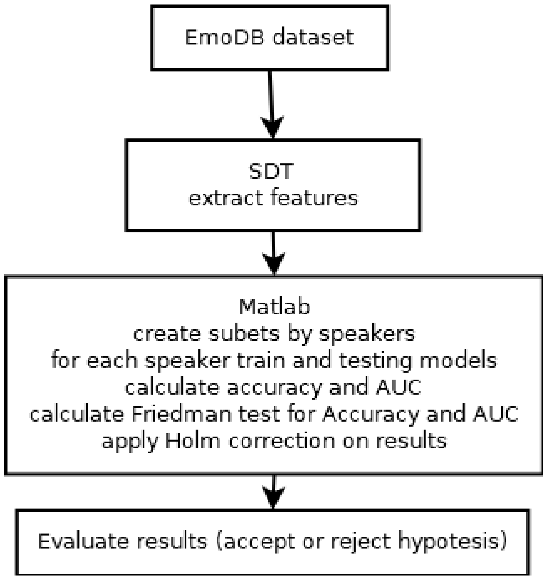

Figure 2.2: An overview of the Friedman test process.

Algorithm 2 STD and Cross validation algorithm for evaluating Friedman test on models

```
procedure SDTANDFRIEDMANTEST
    extract scalar values for audio files from EmoDB with SDT.
    load scalar values items and their classification to Matlab.
    separate them by speaker and create 10 subsets
    for speaker do S_i
        train model on other speakers data
        calculate accuracy from model on S_i data
        calculate AUC for each emotion on S_i data
    save accuracy and AUC results
    calculate Friedman test with Holm correction
    return results
```

MPEG-7 Audio Analyzer and cross-validation (Fig:2.3)

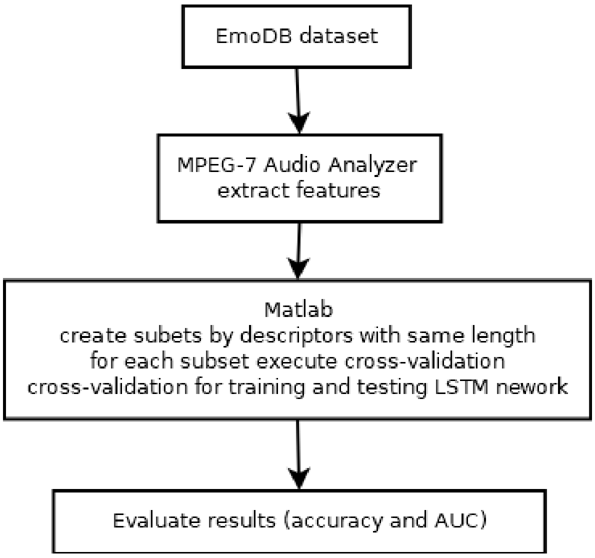

Figure 2.3: An overview of the sentiment analysis by LSTM network process.

Algorithm 3 MAA and Cross validation algorithm for evaluating LSTM network on emotions

```
procedure MAAAndLSTM
    for audio file f do
        load time series T_f
        for i ∈ T_f do
            S_{Len(i)} += i

    for i ∈ S do
        for j ∈ i do
            while Width(j) < MaxWidth(i) do
                newJ += j
            j = newJ[0,MaxWidth(i)]

    for i ∈ S do
        for cross-validation fold F_i do
            train M_{LSTM} on training data not included in F_i
            calculate accuracy from M_{LSTM} on testing data from F_i
            calculate AUC for each emotion on testing data from F_i
    calculate average accuracy and AUC for M_{LSTM}
    save accuracy and AUC results
```

LibROSA and cross-validation

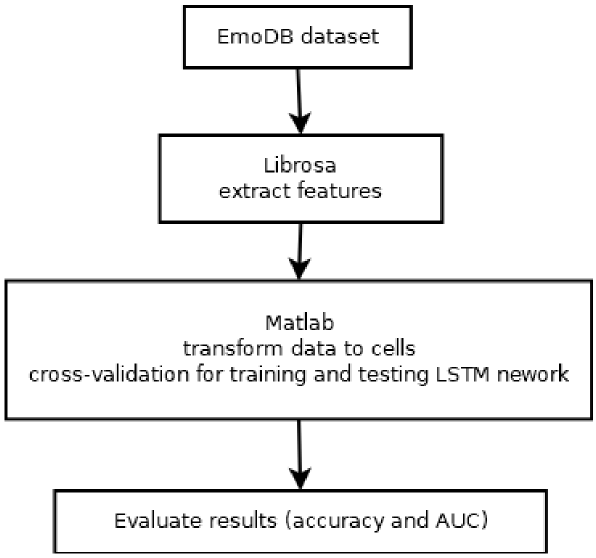

Figure 2.4: An overview of the sentiment analysis by LSTM network process.

Algorithm 4 Librosa and Cross validation algorithm for evaluating LSTM on emotions

procedure LibrosaAndLSTM
  for audio file f **do**
    load time series T_f
    for $i \in T_f$ **do**
      $S+=i$

  for $j \in S$ **do**
    while `Width(`j`)` $<$ `MaxWidth(`S`)` **do**
      $newJ += j$
    $j = newJ[0,$`MaxWidth(`S`)`$]$

  for cross-validation fold F_i **do**
    train M_{LSTM} *on training data not included in* F_i
    calculate accuracy from M_{LSTM} *on testing data from* F_i
    calculate AUC for each emotion on testing data from F_i
  calculate average accuracy and AUC for M_{LSTM}
  save accuracy and AUC results

Chapter 3

Implementation

3.1 Prerequisites

- For SDT: Matlab 2012b and older is needed. (function wavread)
- For LSTM: Matlab 2017b and newer.
- For Librosa: Python package Librosa and Numpy.
- For MAA: working internet connection.

3.2 Python

3.2.1 lstm_features_generator-EmoDB

This file contains the following functions:

- path_to_audiofiles - This function takes path to directory as input parameter and returns python list containing all file names in that directory.
- extract_audio_features - This function takes list of file names and extracts Librosa descriptors (MFCC, Spectral Centroid, Chroma, Spectral Contrast) for each file.

This script saves Librosa descriptors as ndarray (Numpy).

3.2.2 downloader(mechanize).py

This script takes an input and an output folder. For each file in input folder it connects to MAA[32] and performs the remote analysis of MPEG-7 descriptors. The results are downloaded in the form of XML file.

```
│ Python
│  ├── lstm_features_generator-EmoDB ..............................
│  └── downloader(mechanize).py ...................................
└── Matlab
    └── EmoDB
        ├── dt.m ..................................................
        ├── dt1speaker.m ..........................................
        ├── emotion.m .............................................
        ├── generateDataByPython.m ................................
        ├── generateMfcc ..........................................
        ├── generateMpeg7.m .......................................
        ├── generateSubset.m ......................................
        ├── knn.m .................................................
        ├── knn1speaker.m .........................................
        ├── lstm.m ................................................
        ├── meanROC.m .............................................
        ├── mlp.m .................................................
        ├── mlp1speaker.m .........................................
        ├── repeatonmax.m .........................................
        ├── rf.m ..................................................
        ├── rf1speaker.m ..........................................
        ├── ROCbyEmotion.m ........................................
        └── svm.m .................................................
```

3.3 Matlab

3.3.1 Basics and definition

- A cell array is a data type with indexed data containers called cells, where each cell can contain any type of data.[35]
- An ndarray is a (usually fixed-size) multidimensional container of items of the same type and size.[36]

3.3.2 dt.m

"dt.m" loads SDT cell array. Then it trains Decision tree using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.3 dt1speaker.m

"dt1speaker.m" loads SDT cell array sorted by speakers. The current speaker is then taken out from the dataset. The script trains Decision tree on other speakers and evaluates accuracy and AUC for each emotion of current speaker afterwards.

3.3.4 emotion.m

”emotion.m” contains function that translates emotion acronyms from german to english.

3.3.5 generateDataByPython.m

”generateDataByPython.m” converts ndarray (Librosa descriptors) to Matlab cells using function np2mat and saves it.

3.3.6 generateMfcc.m

”generateMfcc.m” reads audio files, then calculates MFCC (via Matlab function) and converts them into one cell array.

3.3.7 generateMpeg7.m

”generateMpeg7.m” transforms XML files (MPEG-7 descriptors) to 7 subsets (identify by length) of cell arrays.

3.3.8 generateSubset.m

”generateSubset.m” loads SDT feature matrices (for each audio file) and converts them into cell array. From SDT cell arrays it creates subset for each speaker that is then used for Friedman test.

3.3.9 knn.m

”knn.m” loads SDT cell array. In the next step trains kNN classifier using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.10 knn1speaker.m

”knn1speaker.m” reads SDT cell array sorted by speakers. The current speaker is taken out from the dataset. The script trains kNN classifier on other speakers and evaluates accuracy and AUC for each emotion of the current speaker afterwards.

3.3.11 lstm.m

”lstm.m” loads time series of MPEG-7 and Librosa. Then trains LSTM network using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.12 meanROC.m

”meanROC.m” creates figure for model. This figure contains ROC curve of 7 emotions and is calculated using mean value from cross-validation.

3.3.13 mlp.m

”mlp.m” reads SDT cell array. In the next step trains Multilayer perceptron using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.14 mlp1speaker.m

”mlp1speaker.m” loads SDT cell array sorted by speakers. The current speaker is then taken out from the dataset. The script in the next step trains Multilayer perceptron on other speakers and evaluates accuracy and AUC for each emotion of current speaker.

3.3.15 repeatonmax.m

”repeatonmax.m” takes each matrix in the cell array and its time sequence is repeated until it matches or exceeds length of the longest matrix in the cell array. Eventual overlap is cut and modified cell array are returned.

3.3.16 rf.m

”rf.m” loads SDT cell array. Then trains Random forest using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.17 rf1speaker.m

”rf1speaker.m” reads SDT cell array, sorted by speakers. The current speaker is taken out from the dataset afterwards. The script then trains Random forest classifier on other speakers and evaluates accuracy and AUC for each emotion of current speaker.

3.3.18 ROCbyEmotion.m

”ROCbyEmotion.m” creates figure for each emotion. This figure contains ROC curve of SDT based classifiers.

3.3.19 svm.m

”svm.m” loads SDT cell array. In the next step trains 7 Support vector machines (one for each emotion) using cross validation for evaluation of accuracy and AUC for each emotion.

3.3.20 svm1speaker.m

”svm1speaker.m” reads SDT cell array, sorted by speakers. The current speaker is then taken out from the dataset. The script trains 7 Support vector machines on other speakers and evaluate accuracy and AUC for each emotion of current speaker afterwards.

Chapter 4

Testing

4.1 Berlin Database of Emotional Speech

For the evaluation of classifiers, we use the publicly available dataset "EmoDB", aka Berlin database of emotional speech. It consists of 535 emotional utterances in 7 emotional categories namely anger, boredom, disgust, fear, happiness, sadness and neutral. These utterances are sentences read by 10 professional actors, 5 males and 5 females [9], which were recorded in an anechoic chamber under supervision by linguists and psychologists. The actors were advised to read these predefined sentences in the targeted emotional categories, but the sentences do not contain any emotional bias. A human perception test was conducted with 20 persons, different from the speakers, in order to evaluate the quality of the recorded data with respect to recognisability and naturalness of presented emotion. This evaluation yielded a mean accuracy 86% over all emotional categories.

4.2 Experimental Testing

4.2.1 Experimental Settings for SDT based classifiers

As input features, the outputs from the Sound Description Toolbox were used. Consequently, the input dimension was 187. The classifiers were compared by means of a 10-fold cross-validation, using the following settings for each of them:

- For the k nearest neighbors classification, the value $k = 9$ was chosen by a grid method from $\langle 1, 80 \rangle$. This classifer was applied to data normalized to zero mean and unit variance.

- Support vector machines are constructed for each of the 7 considered emotions, to classify between that emotion and all the remaining ones. They employ auto-scaled Gaussian kernels and do not use slack variables.

Table 4.1: Accuracy and area under curve (AUC) of the implemented classifiers on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest

Classifier	Accuracy	AUC emotion against the rest		
		Anger	Boredom	Disgust
kNN	0.73	0.956	0.933	0.901
SVM	**0.93**	**0.979**	**0.973**	**0.966**
MLP	0.78	0.977	0.969	0.964
DT	0.59	0.871	0.836	0.772
RF	0.71	0.962	0.949	0.920

Classifier	AUC emotion against the rest			
	Fear	Happiness	Neutral	Sadness
kNN	0.902	0.856	0.962	0.995
SVM	**0.983**	0.904	0.974	**0.997**
MLP	0.969	**0.933**	**0.983**	0.996
DT	0.782	0.683	0.855	0.865
RF	0.921	0.882	0.972	0.992

- The MLP has 1 hidden layer with 70 neurons. Hence, taking into account the input dimension and the number of classes, the overall architecture of the MLP is 187-70-7.

- Classification trees are restricted to have at most 23 leaves. This upper limit was chosen by a grid method from $\langle 1, 50 \rangle$, taking into account the way how classification trees are grown in their Matlab implementation.

- Random forests consist of 50 classification trees, each of them taking over the above restriction. The number of trees was selected by a grid method from 10, 20,…,100.

4.2.2 Comparison of Classifiers for data from SDT

First, we compared the already implemented classifiers on the whole Berlin database of emotional speech, with respect to the accuracy and the area under the ROC curve (area under curve, AUC). Since the ROC curve makes sense only for a binary classifier, we computed areas under 7 separate curves corresponding to classifiers classifying always 1 emotion against the rest. The results are presented in Table 4.1 and in Figure 4.1 4.2. They clearly show SVM as the most promising classifier. It has the highest accuracy, and also the AUC for binary classifiers corresponding to 5 of the 7 classifiers

Then we compared the classifiers separately on the utterances of each of the 10 speakers who created the database. The results are summarized in Table 4.2 for accuracy and Table 4.3 for AUC averaged over all 7 emotions. They indicate a great difference between most of the compared classifiers. This is

Table 4.2: Comparison between pairs of implemented classifiers with respect to accuracy, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the accuracy of the row classifier was higher : on how many parts the accuracy of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal accuracy of all classifiers (significance level 5%), the post-hoc test according to [5, 6] rejects the hypothesis of equal accuracy of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [7]

classifier	kNN	SVM	MLP	DT	RF
kNN		**0:10**	3.5:6.5	9:1	5:5
SVM	**10:0**		10:0	**10:0**	**10:0**
MLP	6.5:3.5	0:10		**10:0**	7:3
DT	1:9	**0:10**	**0:10**		0:10
RF	5:5	**0:10**	3:7	10:0	

confirmed by the Friedman test of the hypotheses that all classifiers have equal accuracy and equal average AUC. The Friedman test rejected both hypotheses with a high significance: With the Holm correction for simultaneously tested hypotheses [7], the achieved significance level (aka p-value) was $4 \cdot 10^{-6}$. For both hypotheses, posthoc tests according to [5, 6] were performed, testing equal accuracy and equal average AUC between individual pairs of classifiers. For the family-wise significance level 5%, they reveal the following Holm-corrected significant differences between individual pairs of classifiers: both for accuracy and averaged AUC: (SVM,DT), (MLP,DT), and in addition between (kNN,SVM), (SVM,RF) for accuracy.

4.2.3 Experimental Settings for LSTM

The output from MPEG-7 Audio Analyzer is set of seventeen descriptors, from these descriptors the subset of following seven descriptor groups that have the same length for each audio file are selected:

- Audio Spectrum Envelope, Audio Spectrum Centroid, Audio Spectrum Spread, Audio Spectrum Projection
- Audio Spectrum Basis
- Audio Spectrum Flatness
- Audio Waveform, Audio Power
- Audio Harmonicity, Audio Fundamental Frequency

Table 4.3: Comparison between pairs of implemented classifiers with respect to the AUC averaged over all 7 emotions, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the AUC of the row classifier was higher : on how many parts the AUC of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal AUC of all classifiers (significance level 5%), the post-hoc test according to [5, 6] rejects the hypothesis of equal AUC of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [7]

classifier	kNN	SVM	MLP	DT	RF
kNN		2:8	0:10	10:0	4:6
SVM	8:2		5:5	**10:0**	9:1
MLP	10:0	5:5		**10:0**	9:1
DT	0:10	**0:10**	**0:10**		0:10
RF	6:4	1:9	1:9	10:0	

- Harmonic Spectral Centroid, Harmonic Spectral Deviation, Harmonic Spectral Spread
- Harmonic Spectral Variation

The following descriptor groups are used as input for LSTM network: MFCC, Spectral Center, Chroma, Spectral Contrast from LibROSA (with settings hop_length=512 and n_mfcc=13), using the following settings:

- The first LSTM network has 1 hidden layer (”last”). Number of neurons was selected from 200, 250,. . . ,400, with 200 epoch.
- The second LSTM network has 2 hidden layers (”sequence”,”last”). Number of neurons was selected from 100,. . . ,250, with 350 epoch.

The number of input neurons depends on MPEG7 group (59,29,19,3,3,3,1) and in case LibROSA is 33. All LSTM networks had fullyConnectedLayer with 7 neurons (the number of classes), softmaxLayer, classificationLayer, were used with training options Adam, mini batch size 350 and were compared with 10-fold cross-validation.

4.2.4 Validation of LSTM Feasibility

Data set is created by a numerical solution of Navier-Stokes[37] differential equation intended for neural networks for CFD (computational fluid dynamics) modeling. It consists of 500 items with 2 input and 6 output sequences. Every sequence has a length of 100. It is a sequence-to-sequence regression.

4.2.4.1 Preprocessing of LSTM netowk

In order to optimize the size of LSTM and computational time the data must be preprocessed by the following algoritm.

```
% input contains Navier-Stokes system of equations
% output contains solution of these equations
% countTimeSeries is number of sacrificed coefficients

numResponses=size(output,3);
for i=1:size(input,1)
    inputx{i}=squeeze(input(i,:,:));
    inputx{i}=inputx{i}(countTimeSeries+1:size(input,2),:);
    outputx{i}=squeeze(output(i,:,:));

    for j=1:numResponses
        for k=1:countTimeSeries
            inputx{i}=[inputx{i} outputx{i}(countTimeSeries+1-k:size(output,2)-k,j)]
        end
    end
    inputx{i}=inputx{i}';
    outputx{i}=outputx{i}(countTimeSeries+1:size(output,2),:)';
end
input=inputx;
output=outputx;
```

4.2.4.2 Experimental Settings

- All sequences are normalized by z-score.[38]
- Number of sacrificed time series is 4.
- The LSTM network has 1 hidden layer ("sequence"). Number of neurons was 200.
- Learning time is 1000 epoch.

LSTM network had fullyConnectedLayer with 6 neurons (the number of output sequences), regressionLayer, were used with training options Adam and a mini batch size 450. They were compared with 10-fold cross-validation.

4.2.4.3 Experimental Result

The LSTM network had an average RMSE 0.0424.

4.2.5 LSTM Classification

4.2.5.1 MPEG-7 Descriptors

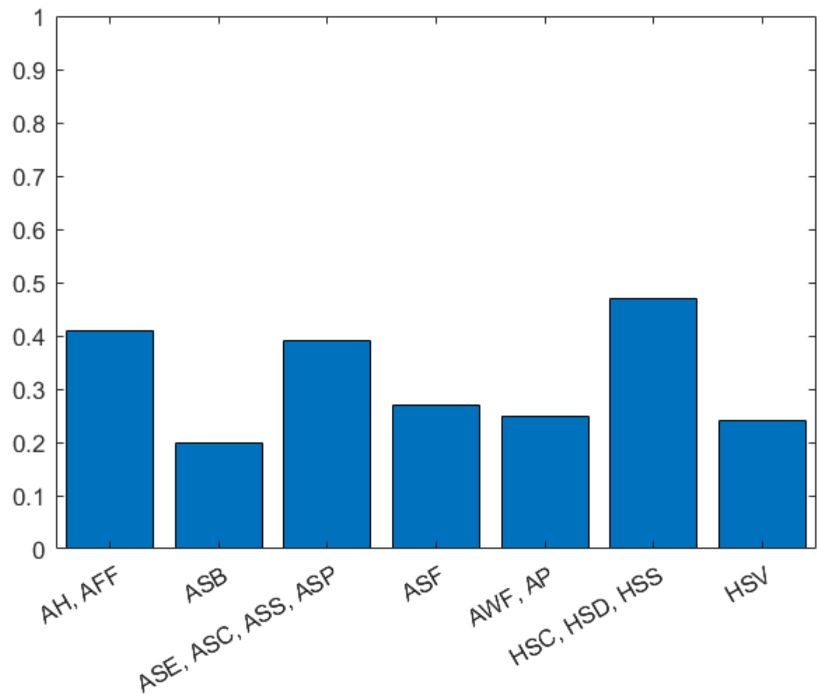

Figure 4.3: Compare accuracy MPEG-7 groups.

For all experiments with outputs from MPEG-7 Audio Analyzer with repeat on max length (for each item, its time sequence is repeated until it matches or exceeds length of the longest item. Eventual overlap is cut) were results for different configurations of LSTM similar: Table 4.4

4.2.5.2 Librosa features

LSTM network with 33 input neurons and 1 hidden layer: Table 4.5

Table 4.4: Accuracy of the LSTM network on MPEG-7 groups

Group	Accuracy
ASE, ASC, ASS, ASP	0.39
ASB	0.20
ASF	0.27
AWF, AP	0.25
AH, AFF	0.41
HSC, HSD, HSS	0.47
HSV	0.24

Table 4.5: Accuracy and area under curve (AUC) of the LSTM with 1 hidden layer on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest

HN	Accuracy	AUC emotion against the rest		
		Anger	Boredom	Disgust
200	0.6464	0.9604	0.8639	0.9224
250	**0.6991**	**0.9639**	0.8931	0.9316
300	0.6952	0.9579	0.8867	0.9256
350	0.6915	0.9494	0.8863	0.9316
400	0.6563	0.9517	**0.9020**	**0.9543**

HN	AUC emotion against the rest			
	Fear	Happiness	Neutral	Sadness
200	0.9405	0.8641	0.9158	0.9860
250	**0.9535**	**0.8975**	**0.9414**	0.9853
300	0.9410	0.8678	0.9291	**0.9932**
350	0.9468	0.8580	0.9179	0.9904
400	0.9208	0.8563	0.9067	0.9879

LSTM network with 33 input neurons and 2 hidden layers: Table 4.6

Table 4.6: Accuracy and area under curve (AUC) of the LSTM with 2 hidden layers on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest

HN	Accuracy	AUC emotion against the rest		
		Anger	Boredom	Disgust
100,100	0.5418	0.9464	0.8494	0.8581
150,100	0.5531	0.9557	0.8422	0.8404
150,150	0.5848	0.9523	0.8513	0.8696
200,100	0.6075	0.9631	0.8595	0.8794
200,150	0.5963	0.9499	0.8294	**0.9168**
200,200	0.6147	0.9480	0.8516	0.8795
250,100	**0.6377**	**0.9655**	**0.8708**	0.8942
250,150	0.6261	0.9621	0.8684	0.8995
250,200	0.6246	0.9592	0.8662	0.9104
250,250	0.5939	0.9419	0.8648	0.8559

HN	AUC emotion against the rest			
	Fear	Happiness	Neutral	Sadness
100,100	0.8664	0.8359	0.8532	0.9660
150,100	0.9052	0.8413	0.8841	0.9609
150,150	0.9100	0.8291	0.8652	0.9622
200,100	**0.9159**	0.8569	0.8804	0.9742
200,150	0.8676	0.8609	0.8649	0.9753
200,200	0.9141	0.8533	0.8714	0.9678
250,100	0.9120	**0.8874**	0.8866	**0.9917**
250,150	0.9097	0.8509	**0.9087**	0.9802
250,200	0.8950	0.8546	0.8863	0.9837
250,250	0.9123	0.8173	0.8834	0.9787

4.3 Evaluation of results

- SVM and MLP are very successfull. SVM has accuracy of over 92% (Table 4.1).
- Statistical testing (Friedman test) confirms differences between SVM, MLP on the one hand, and DT, RF on the other hand (Tables 4.2 4.3).
- MPEG-7 descriptors have different length of time series, therefore they must be separated in subsets with the same length.
- In case of MPEG-7 descriptors, almost every group has the learning function constant (even after 1000 epochs the LSTM network is not able to learn) or the testing accuracy is under 25%.
- Raw MPEG descriptors seem to not be suitable for LSTM networks.
- Aligned time series from MPEG-7 descriptors obtained by repeating shorter sequences have better results than without it (but time points don't match). Figure 4.3.
- For MFCC calculated from Matlab, LSTM network loss function has NaN value. (Probably some internal overflow in the library)
- Aligned time series from Librosa obtained by repeating shorter sequences have better results than without it (but time points didn't match). It is same case with MPEG-7 descriptors.
- Librosa features with aligned length of time series have significantly better results than MPEG-7 descriptors. Accuracy is over 65% (Tables 4.5 4.6).
- LSTM networks work very well on sequence to sequence problems (Subsection 4.2.4).

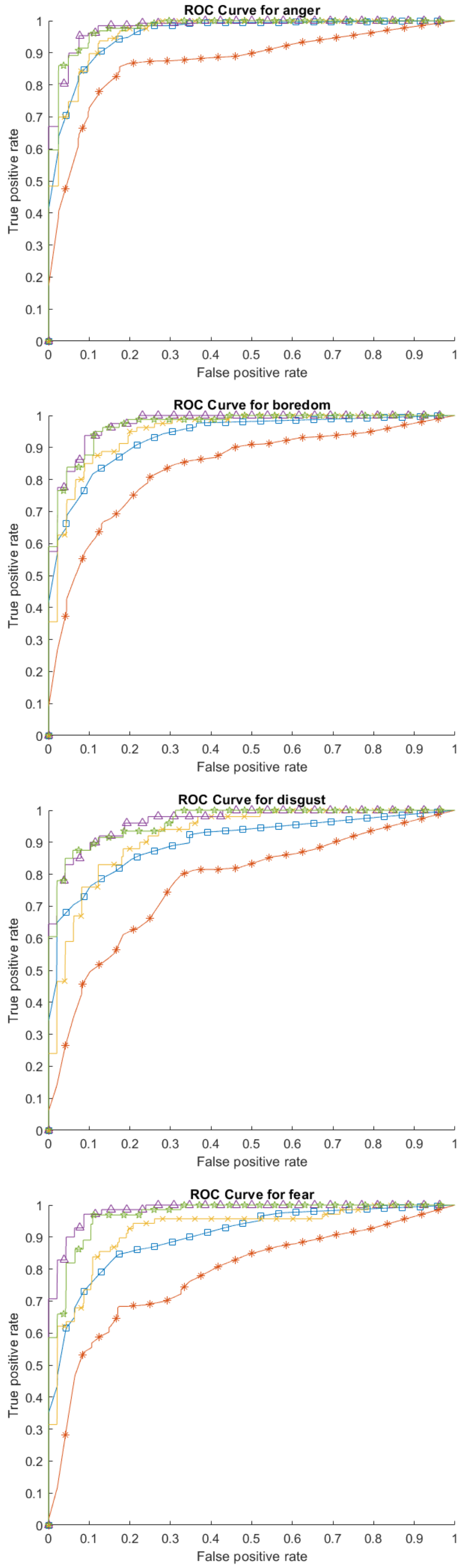

Figure 4.1: ROC curve for all emotions on the whole Berlin database

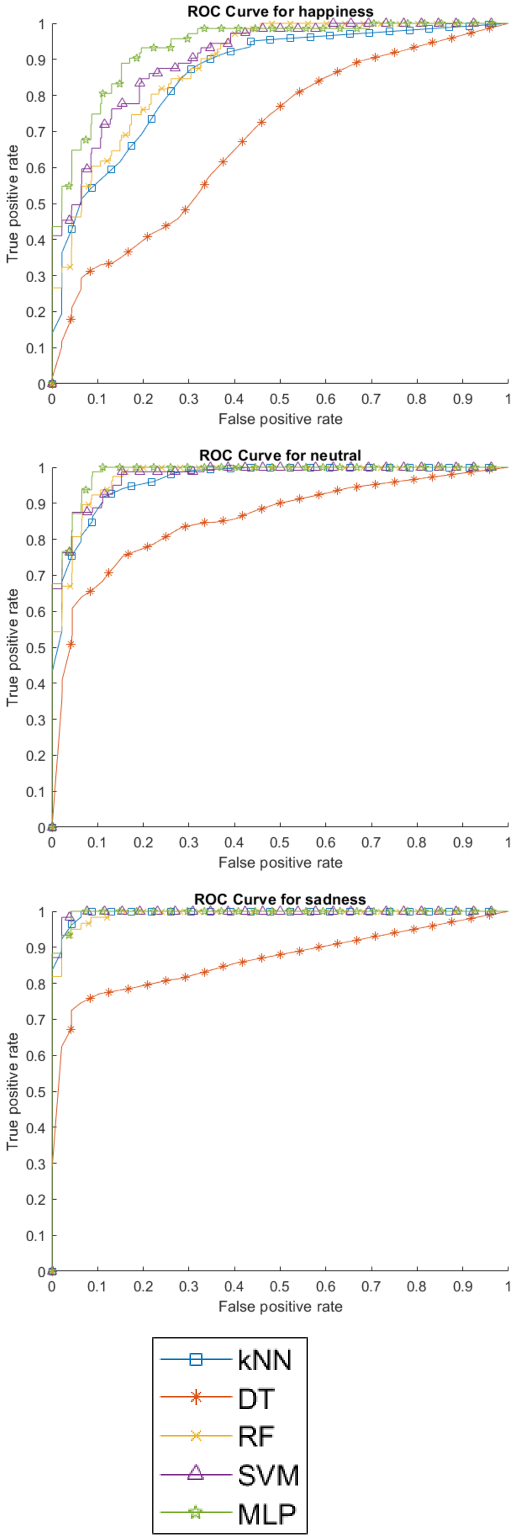

Figure 4.2: ROC curve for all emotions on the whole Berlin database

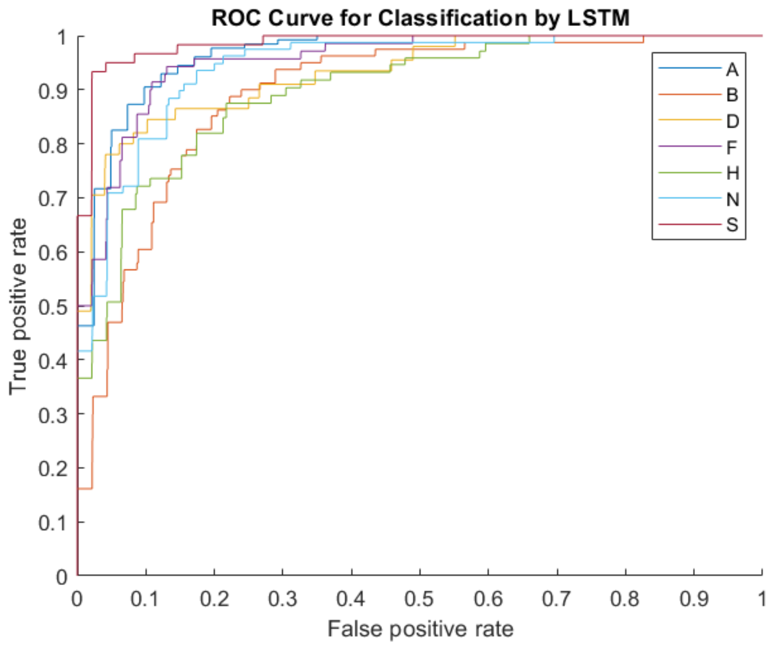

Figure 4.4: ROC curve for LSTM with 1 hidden layer with 250 neurons on the whole Berlin database

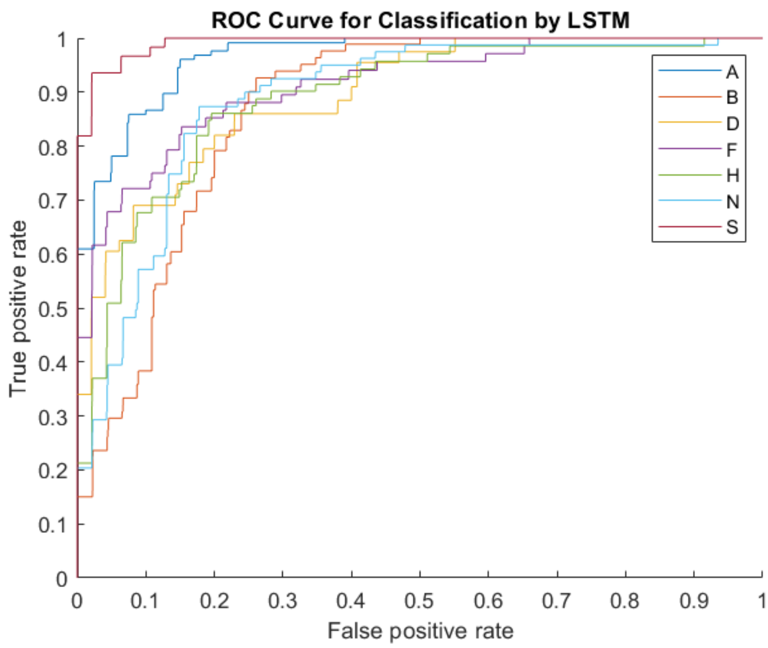

Figure 4.5: ROC curve for LSTM with 2 hidden layer with 250-100 neurons on the whole Berlin database

Conclusion

The presented work investigates the possibilities to analyse emotions in utterances based on MPEG-7 features. We implemented six classification methods, some of them use 187 scalar features and others use time series features. K nearest neighbours classifier, support vector machines, multilayer perceptrons, decision trees and random forests and long short term memory network were implemented.

The obtained results indicate that especially support vector machines and multilayer perceptrons are quite successfull for this task.

Statistical testing confirms significant differences between these two kinds of classifiers on the one hand, and decision trees an random forests on the other hand.

In the beginning of the work, we encountered a problem with LSTM network unability to learn on MPEG-7 audio descriptors. This state was lasting for some months, the extraction of Librosa descriptors was used in meantime to overcome the problem. Librosa was able to achieve only sligthly better results. Via continuous analysis we found out that for satisfactory results the data of approximately the same length should be used. This was not our case due to very variant audio lengths.

After imputation of these ”missing data” by repetition, the LSTM network performance drastically increased. However, the results of classification based on MPEG-7 descriptors (Subsection 4.2.5.1,) was still poor. On the contrary, the Librosa descriptors (Subsection 4.2.5.2) based classification achieved around 70 percent of accuracy.

It can be stated that EmoDB is definitively not an ideal case of dataset for classification by LSTM networks. SVM and MLP (Subsection 4.2.2) outperformed the LSTM by achieving around 80-90% accuracy.

Feasibility of our LSTM network implementation was validated on another dataset, regarding Subsection 4.2.4. This dataset describes regression problem and was considered ideal for LSTM. The network performed very well achieving RMSE around 0.042.

For the training of the classifier, the performance of personal computer was enough. The training of LSTM network in particular was done using a standard GPU. There was no need to use computational power of Metacentrum cloud.

Part of this thesis was presented on the workshop ITAT 18th.[39] .

Future work

For the experimentation with LSTM networks, another dataset also focused on audio classification, GZTAN looks far more appropriate because it contains audio recordings of the same length.

The article[40] suggests that only one LSTM network may not be ideal for multiclass data. This observation indicates that multi model methods such as boosting, bagging or stacking could be good candidates for extending this thesis.

Bibliography

[1] Commons, W. Mpeg7image1. 2007, [Online; accessed December 22, 2018]. Available from: https://en.wikipedia.org/wiki/File:Mpeg7image1.svg

[2] Schultebraucks, L. Introduction to Support Vector Machines. 2017, [Online; accessed December 28, 2018]. Available from: https://medium.com/@LSchultebraucks/introduction-to-support-vector-machines-9f8161ae2fcb

[3] Donges, N. Recurrent Neural Networks and LSTM. 2018, [Online; accessed December 21, 2018]. Available from: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

[4] Commons, W. K-fold cross validation. 2011, [Online; accessed December 21, 2018]. Available from: https://en.wikipedia.org/wiki/File:K-fold_cross_validation.jpg

[5] Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, volume 7, 2006: pp. 1–30.

[6] Garcia, S.; Herrera, F. An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. *Journal of Machine Learning Research*, volume 9, 2008: pp. 2677–2694.

[7] Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, volume 6, 1979: pp. 65–70.

[8] Kim, H.; Moreau, N.; et al. *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval.* John Wiley and Sons, New York, 2005.

[9] Burkhardt, F.; Paeschke, A.; et al. A Database of German Emotional Speech. In *Interspeech*, 2005, pp. 1517–1520.

[10] Lampropoulos, A.; Tsihrintzis, G. Evaluation of MPEG-7 Descriptors for Speech Emotional Recognition. In *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2012, pp. 98–101.

[11] Rauber, A.; Lidy, T.; et al. MUSCLE Network of Excellence: Multimedia Understanding Through Semantics, Computation and Learning. Technical report, TU Vienna, Information and Software Engineering Group, 2004.

[12] Casey, M.; De Cheveigne, A.; et al. MPEG-7 Multimedia Software Resources, 2001, http://mpeg7.doc.gold.ac.uk/.

[13] Lalitha, S.; Madhavan, A.; et al. Speech Emotion Recognition. In *International Conference on Advances in Electronics*, 2014, pp. 92–95.

[14] Xiao, Z.; Dellandrea, E.; et al. Multi-stage Classification of Emotional Speech Motivated by a Dimensional Emotion model. *Multimedia Tools and Applicaions*, volume 46, 2010: pp. 119–145.

[15] L., O. What is Content? Learn from 40+ Definitions, 2013, online at https://www.toprankblog.com/2013/03/what-is-content/.

[16] Hermes, D. J. Sound Perception: The Science of Sound Design. 2015, [Online; accessed September 17, 2015]. Available from: `http://home.ieis.tue.nl/dhermes/lectures/soundperception/04AuditoryFilter.html`

[17] Herrera, P.; Xavier, S. Audio Descriptors and Descriptor Schemes in the Context of MPEG-7. 1999, [Online; accessed December 30, 2018]. Available from: `http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Herrera_1999_ICMC_MPEG7.pdf`

[18] International Organization for Standardization. *Information technology – Multimedia content description interface – Part 1: Systems*. 2002.

[19] International Organization for Standardization. *Information technology – Multimedia content description interface – Part 4: Audio*. 2002.

[20] Lampropoulos, A.; Tsihrintzis, G. *Machine Learning Paradigms: Applications in Recommender Systems*. Intelligent Systems Reference Library, Springer International Publishing, 2015, ISBN 9783319191355. Available from: `https://books.google.cz/books?id=Mb7pCQAAQBAJ`

[21] Yang, Y.-H.; Chen, H. H. *Music Emotion Recognition*. CRC Press, Boca Raton, first edition, 2011, ISBN 9781439850466.

[22] MacKay, D. M. . Psychophysics of perceived intensity: A theoretical basis for Fechner's and Stevens' laws. *Science*, volume 139, 1963: pp. 1213–1216.

[23] Holeňa, M. Basic Concepts Concerning Classification. 2018, [Online; accessed December 25, 2018]. Available from: `https://courses.fit.cvut.cz/MI-ADM/media/lectures/02/kniha1-v1.pdf`

[24] Hastie, T.; Tibshirani, R.; et al. *The Elements of Statistical Learning, 2nd Edition*. Springer, 2008.

[25] Gers, F.; Schmidhuber, J.; et al. Learning to Forget: Continual Prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, 1999, pp. 850–855.

[26] Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. Dissertation thesis, TU München, 2008.

[27] Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Computation*, volume 9, 1997: pp. 1735–1780.

[28] Schölkopf, B.; Smola, A. *Learning with Kernels*. MIT Press, Cambridge, 2002.

[29] OpenML. 10-fold Crossvalidation. [Online; accessed December 31, 2018]. Available from: `https://www.openml.org/a/estimation-procedures/1`

[30] Friedman, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, volume 32, 1937: pp. 675–701.

[31] Wikipedia contributors. Holm–Bonferroni method — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Holm%E2%80%93Bonferroni_method&oldid=878927397`, [Online; accessed February 12, 2019].

[32] Sikora, T.; Kim, H.; et al. MPEG-7-Based Audio Annotation for the Archival of Digital Video, 2003, http://mpeg7lld.nue.tu-berlin.de/.

[33] McFee, B.; McVicar, M.; et al. LibROSA. 2013–2018, [Online; accessed December 30, 2018]. Available from: `https://librosa.github.io/librosa`

[34] Friedman, S. np2mat. 2016. Available from: `https://gist.github.com/samtx/61265a869a04d2c8eee9fe9f314d0c2a`

[35] The MathWorks, I. Cell Arrays. [Online; accessed December 30, 2018]. Available from: `https://www.mathworks.com/help/matlab/cell-arrays.html`

[36] The SciPy, c. The N-dimensional array (ndarray). [Online; accessed December 31, 2018]. Available from: `https://www.numpy.org/devdocs/reference/arrays.ndarray.html`

[37] Weisstein, E. W. Navier-Stokes Equations.From MathWorld–A Wolfram Web Resource. [Online; accessed December 30, 2018]. Available from: `http://mathworld.wolfram.com/Navier-StokesEquations.html`

[38] Kreyszig, E. Advanced Engineering Mathematics (Fourth ed.). Technical report, John Wiley and Sons, New York, 1979, iSBN 0-471-02140-7, 880 pages.

[39] Kožusznik, J.; Holeňa, M.; et al. Sentiment Analysis from Utterances. In *ITAT 2018: Information Technologies – Applications and Theory*, edited by S. Krajči, 2018, pp. 92–99.

[40] Tang, C. P.; Chui, K. L.; et al. Music genre classification using a hierarchical long short term memory (LSTM) model, 2018. Available from: `http://www.cse.cuhk.edu.hk/~khwong/c2018_IWPR2018_LSTM_music_classification.pdf`

Appendix A

Acronyms

AC Accuracy

AFF Audio Fundamental Frequency

AH Audio Harmonicity

ANN Artificial Neural Network

ANOVA Analysis of variance

AP Audio Power

ASB Audio Spectrum Basis

ASC Audio Spectrum Centroid

ASE Audio Spectrum Envelop

ASF Audio Spectrum Flatness

ASP Audio Spectrum Projection

ASS Audio Spectrum Spread

AUC Area Under Curve

AWF Audio Waveform

CFD Computational Fluid Dynamics

CT Classification Trees

EmoDB Berlin database of emotional speech

FM F-measure

FN False Negative

FP False Positive

HSC Harmonic Spectral Centroid

HSD Harmonic Spectral Deviation

HSS Harmonic Spectral Spread

HSV Harmonic Spectral Variation

KKT Karush-Kuhn-Tucker

kNN k Nearest Neighbours

LAT Log Attack Time

LSTM Long short-term memory

MAA MPEG-7 Audio Analyzer

MLP Multilayer Perceptrons

MPEG Moving Picture Experts Group

PR Precision

RF Random Forests

RMSE Root Mean Square Error

RNN Recursive Neural Network

ROC Receiver Operating Characteristic

SDT Sound Description Toolbox

SVM Support vector machine

TC Temporal Centroid

TN True Negative

TP True Positive

Appendix **B**

Contents of CD

```
readme.txt ....................... the file with CD contents description
src ...................................the directory of source codes
├── EmoDB ............................. the directory of dataset EmoDB
├── Matlab ............................the directory of Matlabs scripts
├── Python ............................ the directory of Pythons scripts
└── thesis ............. the directory of LaTeX source codes of the thesis
text .......................................... the thesis text directory
└── DP_Kozusznik_Jiri_2018.pdf ......... the thesis text in PDF format
```