Automatic Multimodal Addressee Detection for Spoken Dialogue Systems

PhD thesis submitted for a degree of Doctor of natural sciences (Dr. rer. nat.)

at Institute of Communications Engineering,
Faculty Engineering, Computer Sciences and Psychology
Ulm University

Submitted by

Oleg Vladimirovich Akhtiamov
from Krasnoyarsk, Russia

Dean:          Prof. Dr.-Ing. Maurits Ortmanns
Scientific Supervisors  Prof. Dr. Dr.-Ing. Wolfgang Minker
(Gutachter):     Prof. Dr. Alexey Karpov
Examiners:     Prof. Dr. Birte Glimm
                Prof. Dr.-Ing. Dr. h.c. Stefan Wesner
Date of Thesis Defence: 24.03.2021

2021
I dedicate this thesis to my wife Lina, our son Artur, and our families.
Acknowledgements. Foremost, I would like to express my deepest gratitude to my supervisors Prof. Dr. Wolfgang Minker and Prof. Dr. Alexey Karpov for sharing their scientific experience with me. I would also like to thank Prof. Dr. Ingo Siegert for a successful scientific cooperation on the mutual topic of interest and especially for sharing the precious data sets with me. I also thank Ulm University and ITMO University for giving me this opportunity to take part in a joint PhD program in cooperation with the aforementioned universities. I also highly appreciate the financial support provided to me by the German Academic Exchange Service (DAAD) and by the Russian Ministry of Science and Higher Education within the "Mikhail Lomonosov" program (projects No. 2.12779.2018/12.2 and No. 8.704.2016/DAAD) and within the program of "Research Grants for Doctoral Candidates and Young Academics and Scientists".
Abstract. Automatic Adressee Detection (AD) is a modern paralinguistics and dialogue challenge that arises in multiparty spoken conversations between several people and a special interlocutor, e.g., a Spoken Dialogue System (SDS) or a person having communicational difficulties. The humans may also talk to each other and even to themselves while interacting with the special interlocutor. In order to follow the conversation and respond if necessary, the interlocutor is supposed to determine whether he/she/it is being addressed by anyone of the conversational partners or not.

AD belongs to paralinguistic problems that are known to be highly language-, corpus-, and even speaker-dependent. There exist several automatic experimental systems capable of predicting addresses given various unimodal (audio-, text-, and video-based) and multimodal signals. Nevertheless, most of these studies are confined to one language (usually English), a single test corpus, and one statement of the classical Human-Machine Addressee Detection (H-M AD) problem. The set of existing studies on multimodal AD is also quite limited. Furthermore, the existing studies on AD lack modern machine learning techniques, e.g., Deep Neural Networks (DNNs) and end-to-end models, that can potentially improve the AD performance.

The aforementioned issues are addressed in the present thesis. We jointly analyse the following corpora: the Voice Assistant Conversation Corpus (VACC) representing a desktop SDS, the SmartWeb Video Corpus (SVC) modelling a portable SDS, the Restaurant Booking Corpus (RBC) representing a call centre SDS, and the HomeBank Child/Adult Addressee Corpus (HB) modelling a passively listening SDS. VACC and SVC are dedicated to similar problems of classical H-M AD. RBC represents a slightly different problem of complexity-identical H-M AD. HB models another problem of Human-Human Addressee Detection (H-H AD). The latter corpus contains English speech, while the others were uttered in German. Using the four corpora, we carry out a cross-corpus study on acoustic and textual AD to determine differences and similarities between the various AD problem statements, SDS scenarios, corpora, and domains. We additionally specify several corpus-specific and corpus-independent aspects of AD, e.g., a set of common relevant acoustic features, the optimal length of acoustic and verbal context, the most typical lexical units for each addressee category, etc. These results allow us to improve the AD performance on the considered corpora.

We propose several architectures of acoustic and textual classifiers based on advanced machine learning techniques, including DNNs and end-to-end models. Employing the proposed individual classifiers and their fusion, we managed to outperform the existing AD baselines on the considered corpora.

We introduce a novel approach to cross-corpus data augmentation, based on the mixup method, that regularises our DNNs dealing with acoustic features and not just improves their single-task performance on a specific corpus, but also enables effective multitask learning on several corpora at once. Moreover, the corpora being leveraged for the data augmentation may be collected for various (but similar) tasks and even uttered in different languages.

We also analyse users’ speech, recognised text, and video regarding the classical H-M AD problem of SVC by means of machine learning techniques. We propose various methods for modelling different components of each modality, such as prosody, lexical content, syntax, semantic content, and visual focus of attention. We estimate their relative contributions to the total AD performance and also perform their fusion. Additionally, we investigate the performance of the proposed models on speech categories of various spontaneity levels.
# Contents

1 Introduction

1.1 Solutions for Improving the Adaptivity of Spoken Dialogue Systems .......................... 1

1.2 Problem of Human-Machine Addressee Detection ..................................................... 2

1.2.1 Special Case: Classical Human-Machine Addressee Detection .............................. 3

1.2.2 Special Case: Complexity-Identical Human-Machine Addressee Detection ............... 5

1.2.3 Special Case: Human-Human Addressee Detection ............................................... 5

1.3 Motivation ................................................. 6

1.3.1 Role of Addressee Detection for Spoken Dialogue Systems ............................... 6

1.3.2 Joint Analysis of Various Statements of the Addressee Detection Problem .......... 9

1.4 Contribution ........................................... 9

1.5 Outline ................................................. 10

2 Background and Related Work .......................................................... 13

2.1 Related Work .............................................................................. 13

2.1.1 "Smart Web" and Other Multimodal Studies ....................................................... 13

2.1.2 "Conversational Browser" Studies ................................................................. 16

2.1.3 Studies of Deep Neural Networks ................................................................. 18

2.2 Feature Extraction ........................................................................ 19

2.2.1 Acoustic Features ......................................................................................... 20

2.2.2 Beamforming Features ............................................................................... 25

2.2.3 ASR Features ............................................................................................... 26

2.2.4 ASR Metafeatures ....................................................................................... 41

2.2.5 Visual Features ........................................................................................... 42

2.2.6 System Features ........................................................................................... 44

2.3 Feature Selection and Feature Transformation Methods ................................. 44

2.3.1 Principal Component Analysis ....................................................................... 44

2.3.2 Recursive Feature Elimination ....................................................................... 46

2.3.3 Intramodel Dimensionality Reduction ............................................................. 46

2.4 Classification Models ........................................................................... 47

2.4.1 Support Vector Machine .............................................................................. 47

2.4.2 K-Nearest Neighbours .................................................................................. 53

2.4.3 Nearest Centroid Classifier .......................................................................... 54

2.4.4 Artificial Neural Network .............................................................................. 55

2.5 Fusion ............................................................................................... 66

2.5.1 Data-Level Fusion ......................................................................................... 66

2.5.2 Feature-Level Fusion ..................................................................................... 66
## 3 Corpora and Tools

<table>
<thead>
<tr>
<th>Subchapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 SmartWeb Video Corpus</td>
<td>69</td>
</tr>
<tr>
<td>3.1.1 Experimental Setup</td>
<td>69</td>
</tr>
<tr>
<td>3.1.2 Data Collection</td>
<td>70</td>
</tr>
<tr>
<td>3.1.3 Data Annotation</td>
<td>72</td>
</tr>
<tr>
<td>3.2 Voice Assistant Conversation Corpus</td>
<td>75</td>
</tr>
<tr>
<td>3.2.1 Corpus Design</td>
<td>75</td>
</tr>
<tr>
<td>3.2.2 Recording Setup</td>
<td>76</td>
</tr>
<tr>
<td>3.2.3 Perceptual Experiments</td>
<td>76</td>
</tr>
<tr>
<td>3.2.4 Corpus Characteristics</td>
<td>77</td>
</tr>
<tr>
<td>3.3 Restaurant Booking Corpus</td>
<td>79</td>
</tr>
<tr>
<td>3.3.1 Corpus Design</td>
<td>79</td>
</tr>
<tr>
<td>3.3.2 SDS Simulation</td>
<td>80</td>
</tr>
<tr>
<td>3.3.3 Recording Setup</td>
<td>80</td>
</tr>
<tr>
<td>3.3.4 Corpus Characteristics</td>
<td>80</td>
</tr>
<tr>
<td>3.3.5 Perceptual Experiments</td>
<td>82</td>
</tr>
<tr>
<td>3.3.6 Computational Experiments</td>
<td>82</td>
</tr>
<tr>
<td>3.4 HomeBank Child/Adult Addressee Corpus</td>
<td>83</td>
</tr>
<tr>
<td>3.4.1 Corpus Design</td>
<td>83</td>
</tr>
<tr>
<td>3.4.2 Data Collection</td>
<td>83</td>
</tr>
<tr>
<td>3.4.3 Data Annotation</td>
<td>84</td>
</tr>
<tr>
<td>3.5 Tools and Computational Infrastructure</td>
<td>85</td>
</tr>
<tr>
<td>3.5.1 Acoustic Feature Extraction</td>
<td>85</td>
</tr>
<tr>
<td>3.5.2 ASR Feature Extraction</td>
<td>85</td>
</tr>
<tr>
<td>3.5.3 ASR Metafeature Extraction</td>
<td>86</td>
</tr>
<tr>
<td>3.5.4 Visual Feature Extraction</td>
<td>86</td>
</tr>
<tr>
<td>3.5.5 Feature Selection and Feature Transformation</td>
<td>86</td>
</tr>
<tr>
<td>3.5.6 Classification</td>
<td>86</td>
</tr>
<tr>
<td>3.5.7 Computational Infrastructure</td>
<td>86</td>
</tr>
<tr>
<td>3.6 Summary</td>
<td>86</td>
</tr>
</tbody>
</table>

## 4 Speech and Text Analysis for Cross-Corpus Addressee Detection

<table>
<thead>
<tr>
<th>Subchapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Proposed Classifiers</td>
<td>90</td>
</tr>
<tr>
<td>4.1.1 ComParE_func Classifier</td>
<td>90</td>
</tr>
<tr>
<td>4.1.2 ComParE_LLD Classifier</td>
<td>92</td>
</tr>
<tr>
<td>4.1.3 ASR_meta Classifier</td>
<td>92</td>
</tr>
<tr>
<td>4.1.4 Semantic Classifier</td>
<td>94</td>
</tr>
<tr>
<td>4.2 Metrics</td>
<td>99</td>
</tr>
<tr>
<td>4.3 Data Splitting</td>
<td>100</td>
</tr>
<tr>
<td>4.4 Speech Analysis</td>
<td>101</td>
</tr>
<tr>
<td>4.4.1 Keep-One-Corpus-In Experiments</td>
<td>101</td>
</tr>
<tr>
<td>4.4.2 Acoustic Feature Selection Experiments</td>
<td>103</td>
</tr>
<tr>
<td>4.4.3 Experiments with Various Acoustic Context Lengths</td>
<td>106</td>
</tr>
<tr>
<td>4.4.4 Experiments with the ASR Metafeatures</td>
<td>110</td>
</tr>
</tbody>
</table>
X  Contents

List of Own Publications ................................................................. 186
List of Figures ............................................................................... 187
List of Tables ................................................................................. 190
Acronyms

AD  Adressee Detection
ANN  Artificial Neural Network
API  Application Programming Interface
ASR  Automatic Speech Recogniser
ATT  Attractiveness

BAS  Bavarian Archive for Speech Signals
BLSTM  Bidirectional Long Short-Term Memory
BOW  Bag-of-Words

CB  Conversational Browser
CNN  Convolutional Neural Network
ComParE  The Interspeech Computational Paralinguistics Challenge
CPU  Central Processing Unit
CRNN  Convolutional Recurrent Neural Network
CW  Confident Weights

DCT  Discrete Cosine Transform
DET  Detection Error Tradeoff
DNN  Deep Neural Network

E2E  End-to-End
EBP  Error Back Propagation Algorithm
EEG  Electroencephalogram
EER  Equal Error Rate

FFT  Fast Fourier Transform
FPS  Frames per Second
FT  Feature Transformation Method Based on Terms Belonging to Classes

GloVe  Global Vectors
GMM  Gaussian Mixture Model
GPU  Graphics Processing Unit
GR  Gain Ratio
GUI  Graphical User Interface
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-H</td>
<td>Human-Human</td>
</tr>
<tr>
<td>H-H AD</td>
<td>Human-Human Addressee Detection</td>
</tr>
<tr>
<td>H-H-M</td>
<td>Human-Human-Machine</td>
</tr>
<tr>
<td>H-M</td>
<td>Human-Machine</td>
</tr>
<tr>
<td>H-M AD</td>
<td>Human-Machine Addressee Detection</td>
</tr>
<tr>
<td>HB</td>
<td>HomeBank Child/Adult Addressee Corpus</td>
</tr>
<tr>
<td>HNR</td>
<td>Harmonic-to-Noise Ratio</td>
</tr>
<tr>
<td>HQ</td>
<td>Hedonistic Quality</td>
</tr>
<tr>
<td>HQ-I</td>
<td>Hedonistic Quality - Identity</td>
</tr>
<tr>
<td>HQ-S</td>
<td>Hedonistic Quality - Stimulation</td>
</tr>
<tr>
<td>IDF</td>
<td>Inverse Document Frequency</td>
</tr>
<tr>
<td>IOT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IQ</td>
<td>Interaction Quality</td>
</tr>
<tr>
<td>IS13_ComParE</td>
<td>The Interspeech 2013 Computational Paralinguistics Challenge</td>
</tr>
<tr>
<td>k-NN</td>
<td>k Nearest Neighbours</td>
</tr>
<tr>
<td>KOCI</td>
<td>Keep-One-Corpus-In</td>
</tr>
<tr>
<td>LDC</td>
<td>Linear Discriminant Classifier</td>
</tr>
<tr>
<td>LLD</td>
<td>Low-Level Descriptor</td>
</tr>
<tr>
<td>LLR</td>
<td>Linear Logistic Regression</td>
</tr>
<tr>
<td>LM</td>
<td>Language Model</td>
</tr>
<tr>
<td>LOCO</td>
<td>Leave-One-Corpus-Out</td>
</tr>
<tr>
<td>LOSGO</td>
<td>Leave-One-Speaker-Group-Out</td>
</tr>
<tr>
<td>LOSO</td>
<td>Leave-One-Speaker-Out</td>
</tr>
<tr>
<td>LPC</td>
<td>Linear Predictive Coding</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel-Frequency Cepstral Coefficient</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>NOT</td>
<td>On-Talk</td>
</tr>
<tr>
<td>NTW</td>
<td>Novel Term Weighting</td>
</tr>
<tr>
<td>Off-Talk</td>
<td>Human-directed speech</td>
</tr>
<tr>
<td>Off-View</td>
<td>A user visually focused on a person/object other than the spoken dialogue system</td>
</tr>
<tr>
<td>On-Talk</td>
<td>Machine-directed speech</td>
</tr>
<tr>
<td>On-View</td>
<td>A user visually focused on the spoken dialogue system</td>
</tr>
<tr>
<td>OOT</td>
<td>Other Off-Talk</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation</td>
</tr>
<tr>
<td>POS</td>
<td>Part-of-Speech</td>
</tr>
<tr>
<td>POT</td>
<td>Paraphrased Off-Talk</td>
</tr>
<tr>
<td>PQ</td>
<td>Pragmatic Quality</td>
</tr>
<tr>
<td>PTT</td>
<td>Push-to-Talk</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>RASTA</td>
<td>Relative Spectral Analysis</td>
</tr>
<tr>
<td>RBC</td>
<td>Restaurant Booking Corpus</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RF</td>
<td>Relevance Frequency</td>
</tr>
<tr>
<td>RFE</td>
<td>Recursive Feature Elimination</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>ROT</td>
<td>Read Off-Talk</td>
</tr>
<tr>
<td>SDS</td>
<td>Spoken Dialogue System</td>
</tr>
<tr>
<td>SH</td>
<td>Smart Home</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>SitPro</td>
<td>Situational Prompting</td>
</tr>
<tr>
<td>SOT</td>
<td>Spontaneous Off-Talk</td>
</tr>
<tr>
<td>SVC</td>
<td>SmartWeb Video Corpus</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>T2V</td>
<td>Text-to-Vector</td>
</tr>
<tr>
<td>TF</td>
<td>Term Frequency</td>
</tr>
<tr>
<td>TM2</td>
<td>Term Second Moment</td>
</tr>
<tr>
<td>TRR</td>
<td>Term Relevance Ratio</td>
</tr>
<tr>
<td>TS</td>
<td>Technical System</td>
</tr>
<tr>
<td>TUI</td>
<td>Textual User Interface</td>
</tr>
<tr>
<td>TWR</td>
<td>Trigger-Word Recognition</td>
</tr>
<tr>
<td>UAP</td>
<td>Unweighted Average Precision</td>
</tr>
<tr>
<td>UAR</td>
<td>Unweighted Average Recall</td>
</tr>
<tr>
<td>VACC</td>
<td>Voice Assistant Conversation Corpus</td>
</tr>
<tr>
<td>VJ</td>
<td>Viola-Jones Method</td>
</tr>
<tr>
<td>W2V</td>
<td>Word-to-Vector</td>
</tr>
<tr>
<td>WCDMA</td>
<td>Wideband Code Division Multiple Access</td>
</tr>
<tr>
<td>WER</td>
<td>Word Error Rate</td>
</tr>
<tr>
<td>WOZ</td>
<td>Wizard-of-Oz Dialogue Simulation</td>
</tr>
</tbody>
</table>
Automatic Spoken Dialogue Systems (SDSs) appeared just a couple of decades ago and have already become part of our everyday life. Their growing popularity can easily be explained: speech is the most natural way of communication between people, and therefore they usually prefer speech-based user interfaces over textual (TUI) and graphical input (GUI) alone when it comes to natural interaction with technical systems [Kinsella, 2018]. Speech input alone is particularly convenient for specifying abstract actions or defining multiple entities [Billinghurst, 2013]. Sometimes a combination of speech and additional modalities, usually touch or gesture, is preferred to resolve deictic references or spatial information [Tse et al., 2006; Bubalo et al., 2016]. SDSs are thus considered to be the most user-friendly Human-Machine (H-M) interfaces nowadays. However, a higher degree of user-friendliness results in more problems for system developers. Spoken data input is essentially more diverse than textual or graphical one and contains paralinguistic information besides verbal linguistic content. The word "paralinguistic" came from Greek and Latin and means "beside language" or "alongside linguistic". Paralinguistics thus does not concern the typical phenomena of a language; it deals with how someone says something rather than what they say. Age, gender, speaker state, speaker trait, speaker focus of attention, etc. - these factors belonging to paralinguistic phenomena influence humans’ speech dramatically. Nevertheless, the problem of processing the speech diversity has already been solved with highly efficient Automatic Speech Recognisers (ASRs) based on Deep Neural Networks (DNNs). Such ASRs are capable of recognising speech under various environmental conditions at an accuracy level near to human beings. The paralinguistic data filtered out from the verbal content does not play a critical role in problem solving, though this data contains valuable user information that we can leverage in order to make our SDS more flexible and human-like.

1.1 Solutions for Improving the Adaptivity of Spoken Dialogue Systems

The capabilities of modern SDSs are not confined only to the range of tasks that such systems can solve. It is intuitively clear that any SDS is supposed to take humans’ behaviour as the pattern for its own actions. As any human being, the system should adapt to its interlocutors in several aspects, such as their manner of speech, emotional state, and focus of attention. Many research efforts are thus focused on improving SDSs’ adaptability to users’ states and traits and on modelling human-
like behaviour in general. Ultes et al. [2016] suggested a user-centred dialogue manager that can change its strategy in a problem-oriented dialogue according to the level of a user’s satisfaction. Pragst and Ultes [2018] developed an approach to directness estimation for SDSs. The approach determines a desired degree of system directness that a user wants to get from the system. Depending on cultural background and a politeness level, some people feel more comfortable when they and their interlocutors formulate their thoughts indirectly rather than acting in a straightforward way to solve a certain task. Raveh et al. [2018] presented a study on mutual speech variation influences in a human-computer dialogue. The system developed by the researchers is capable of detecting and tracking the state of phonetic features in a user’s speech and adapting accordingly so that the system’s manner of speech is comfortable for the user. Tsai et al. [2015] designed a multimodal approach allowing an SDS to classify users’ utterances as human-directed or machine-directed and to react explicitly to the latter ones only. The system possessing this function can maintain a multiparty spoken conversation with several users in a realistic way. This system solves a typical Addressee Detection (AD) problem that is the topic of the present thesis.

1.2 Problem of Human-Machine Addressee Detection

Traditionally, the interaction between a human and an SDS is considered as a pure H-M conversation with no third-party interlocutors as shown in Figure 1.1, i.e., all the speech captured by the system microphone is assumed to be system-directed [Paek et al., 2000]. However, this is not always the case in real-life applications. A real system may face a mixed conversation involving several (usually two or more) humans talking to the system, to each other, and even to themselves from time to time. Such a mixed interaction is depicted in Figure 1.2 and will further be denoted as Human-Human-Machine (H-H-M) interaction. H-H-M interaction appears in cooperative tasks, e.g., several people are negotiating how they want to spend this evening, asking the SDS to show information about various cafes, cinema, night clubs, etc., and discussing possible alternatives. The system is supposed to detect machine-directed utterances that should be processed in an explicit way with a direct system response. To human-directed utterances, the system should not give any direct response or interfere in the conversation in any way. Instead, the SDS can process such utterances in an implicit way to extract some additional information about the speakers, e.g., their emotions in order to adapt to them, or about the conversation course in order to stay up to date on the process of problem solving and behave proactively. The system’s proactive behaviour can improve users’ experience, e.g., by optimising the waiting time when the SDS automatically downloads some preliminary information on the subject that the users just mentioned in their discussion between each other.

The described problem of distinguishing machine-directed utterances (On-Talk) from human-directed ones (Off-Talk) is called Human-Machine Addressee Detection (H-M AD). Another term that appears in scientific literature is Off-Talk detection.

How can the system "guess" that it is being addressed? First, it can do this, analysing the verbal content of users’ speech. The underlying idea for this approach is that people address the others either explicitly, specifying desirable addressees by names, or implicitly, using contextual markers, e.g., domain-related lexical units [S. V. Ravuri and Stolcke, 2014] or contextual references [Ouchi and Tsuboi, 2016]. We denote the AD problem for text as textual or text-based AD for simplicity.
1.2 Problem of Human-Machine Addressee Detection

Second, humans tend to change their normal manner of speech (prosody), making it more rhythmical, louder, and generally more understandable, as soon as they start talking to the SDS since they do not perceive it as an adequate conversational partner [Shriberg et al., 2012; Siegert and Krüger, 2018]. We denote the AD problem for audio as acoustic or audio-based AD for simplicity. Third, people tend to look at the object or person they are talking to [Maglio et al., 2000; Skantze and Gustafson, 2009]. We denote the AD problem for video as visual or video-based AD for simplicity. Most of the existing works on H-M AD are based on at least one of these three concepts, while the most thorough studies include several of them at once [Batliner, Hacker, and Nöth, 2008; Tsai et al., 2015]. We denote the AD problem for several modalities as multimodal AD for simplicity. There also exist exotic approaches to H-M AD, e.g., electroencephalogram (EEG) analysis [Haider et al., 2018; Haider et al., 2019].

1.2.1 Special Case: Classical Human-Machine Addressee Detection

In order to simplify the experimental setup and extend the set of corpora that can potentially be utilised to model the H-M AD problem, most researchers assume that only Human_1 from Figure 1.2 interacts with the system as a user. Human_2 acts as a companion discussing the information obtained from the SDS. The companion assists the user without talking to the system directly. This
assumption is reflected in Figure 1.3. The SDS captures the user’s speech that may be directed either to the system or to the companion, and this is exactly the place where addressee ambiguity arises. An example of a simplified H-H-M conversation is given in Figure 1.4. The smartphone SDS introduced in this example is controlled by a single human (user). The system is guiding the people through the city of Berlin and answering their questions about various sights.

Most existing studies on H-M AD were conducted according to one of the interaction schemes depicted in Figure 1.2 and 1.3 [Batliner, Hacker, and Nöth, 2008; Shriberg et al., 2013; Tsai et al., 2015; Mallidi et al., 2018; Siegert, Krüger, et al., 2018]. Let us denote these special cases as classical H-M AD. The experiments were designed in such a way that the system and the human companion always played different dialogue roles, e.g., the SDS tended to interact with the user passively, only responding to the user’s queries, while the companion could behave actively, initiating his or her interaction with the user. Moreover, the H-M interaction was simpler and also more problem-
oriented than the H-H one, resulting in essential lexical and grammatical differences between the H-M and the H-H dialogue domains. Furthermore, the effect of a visible counterpart arose since the user could see both conversational partners simultaneously. M. K. Lee et al. [2010] investigated this effect and noted that the way people were talking to a technical system depended on how it looked like, i.e., whether the SDS had a human-like appearance or not. As a result, these factors can potentially influence speakers’ behaviour, leading to more obvious addressee patterns.

### 1.2.2 Special Case: Complexity-Identical Human-Machine Addressee Detection

Siegert et al. [2019] proposed a novel approach to data collection to eliminate most of the aforementioned factors. The only remaining factor is speakers’ explicit awareness of their interlocutor’s nature (technical system or human being). Experimental data is collected according to an interaction scheme presented in Figure 1.5. Let us denote this special case as complexity-identical H-M AD. A user solves a certain task with a call centre employee (Figure 1.5a), and then the user accomplishes a similar task with an SDS (Figure 1.5b) so that the employee and the system play the same dialogue role. The first and the second task have similar complexity and content but different wording that excludes word-for-word repetitions. The effect of a visible counterpart does not appear since the interlocutors do not see each other during a phone call. The main question that arises after stating the classical and the complexity-identical H-M AD problem is whether these problem statements complement one another in the sense of speakers’ behaviour or not. We believe that the joint analysis of the two aforementioned problem statements can shed light on more general behavioural patterns that people use to address machines.

![Fig. 1.5: Complexity-identical H-M AD.](image)

### 1.2.3 Special Case: Human-Human Addressee Detection

There may also be a human interlocutor instead of the machine in Figure 1.2 and 1.3. Let us denote this special case as Human-Human Addressee Detection (H-H AD). The speakers tend to emphasise and simplify their speech directed to the SDS, as if they were talking to a human experiencing communicational difficulties, e.g., a hard-of-hearing person [Batliner, Hacker, and Nöth, 2008], a person suffering from psycho- or neurological disorders, or a child having a lack of communicational...
experience [Schuller et al., 2017]. The speakers accommodate their lexicon and manner of speech, e.g., employing a more regular rhythmic up-and-down energy pattern, to such special interlocutors in order to attract their attention and also to make the utterances easier to interpret [Shriberg et al., 2012]. Similarly to classical H-M AD, the H-H AD problem arises in conversations between a group of people and a special interlocutor, e.g., a child. An example of a conversation between two adults and a child is depicted in Figure 1.6 (compare with Figure 1.4). In this example, our system is supposed to distinguish between child-directed and adult-directed utterances. We consider both H-M AD and H-H AD binary classification problems as one, i.e., the utterances belonging to the first category are directed to a special addressee that may be an SDS or a child. The utterances belonging to the second category are directed to ordinary adults without any impairments that might cause miscommunication. The main question that arises after stating the classical H-M AD and the H-H AD problem is whether these problem statements complement one another in the sense of speakers’ behaviour or not. We believe that the joint analysis of the two aforementioned problem statements can shed light on more general behavioural patterns that people use to address special interlocutors.

1.3 Motivation

1.3.1 Role of Addressee Detection for Spoken Dialogue Systems

People possess the ability to detect addressees within spoken conversations that allows them to maintain multiparty interactions involving several conversational partners. In other words, humans "extract" some multimodal features from a spoken conversation; this information enables each interlocutor to determine whether he or she is being addressed or not. AD is extremely useful for SDSs, such as personal assistants, social robots, and chat bots, and can essentially improve their performance alongside with users’ perception of such systems. Virtual assistants, e.g., Siri, Cortana, Alexa, Google Assistant, and Alisa, are typical examples of modern SDSs. However, none of the aforementioned off-the-shelf systems implements the automatic AD function. Instead, they employ some of the following three quite straightforward strategies to avoid any addressee ambiguity during spoken input. The first solution is a patch rather than a real strategy and implies rejecting users’
utterances in case of failure-to-recognize or failure-to-interpret [Pack et al., 2000; Dowding et al., 2006]. The second solution called Push-to-Talk (PTT) is manual and widespread in other spoken interfaces besides SDSs, e.g., in two-way radio devices. In order to let the system know that it is about to be addressed, a user needs to push a button directly before talking to the SDS. This solution was applied to systems belonging to previous generations. Systems of the current generation use the third solution called Trigger-Word Recognition (TWR). Highly accurate modern ASRs allow the SDSs to recognise trigger-words in continuous speech in the wild. The system is continuously listening to humans’ speech and activates as soon as they pronounce a specific word, which is usually the system’s name. This approach is more natural for humans than PTT, though they tend to use the name-based addressing only in the first turn of a conversation thread. In subsequent turns, people usually specify addresses in the conversational context without calling them by name. Therefore, the main disadvantage of TWR is a lack of flexibility (the set of predefined trigger-words is usually quite limited) that leads to unnatural interaction (users have to call the system by name each time).

All the existing solutions to handle addressee ambiguity are no longer applicable since they do not correspond to the main trend of SDS development aimed at reaching a higher degree of naturalness for H-M conversations and bringing them as close as possible to H-H conversations. Furthermore, the system remains inactive most of the operational time. The limitations of the existing AD techniques may sometimes result in very amusing and tricky situations [Liptak, 2017]. Therefore, the necessity of advanced AD methods based on machine learning (ML) arises in modern spoken applications.

Being focused on SDS quality improvement through introducing an AD component, we need to specify a conceptual scheme of a system and the place for the component in it. This general scheme is depicted in Figure 1.7 and represents the minimal set of mandatory modules included in any SDS. A user’s speech is captured by the microphone of an ASR component. The recognised text undergoes the stage of text analysis where it is transformed into a semantic representation in a certain vector space. The obtained semantic representation of the utterance goes through a dialogue manager interacting with the target application and receiving a response from it. After that, a text generator transforms the application’s response into a textual message in a natural language. An automatic speech synthesiser generates a spoken response based on this message and transmits it back to the user.

In order to enable our system to detect addressees automatically, we add an AD component to it as illustrated in Figure 1.8. The component receives the extracted acoustic features and recognised text from the speech recogniser alongside with the user’s head position and face coordinates obtained from a face recogniser. Performing data fusion, the AD module processes the acoustic, textual, and visual information and returns a prediction regarding the user’s focus of attention. This prediction is used as additional input to the dialogue manager so that it can make a decision whether the user’s utterance should be directly responded to or not and change the dialogue strategy accordingly.

The AD component is beneficial for the following applications:

- **H-M Interaction.** Introducing the AD module will make SDSs truly social. Virtual assistants will obtain the ability to maintain multiparty conversations involving several conversational partners. Such a system will become part of a collective rather than an alien element. As a result, humans will be able to focus specifically on problem solving with the SDS rather than on accom-
Fig. 1.7: Conceptual scheme of any SDS.

Fig. 1.8: The place for an AD component in the SDS architecture.

modating their behaviour to the system. In general, the AD function will accelerate and facilitate the process of cooperative problem solving.

- **Assistive Living Environments.** Users do not want to burden themselves with extra communication efforts when it comes to interaction with household appliances. Due to the AD function, advanced ambience control by means of voice will become more intuitive within smart environments, such as Smart Home (SH) and Internet of Things (IOT). Smart devices will be able to "guess" when speakers are addressing them even without using any trigger-words.
• **H-H Interaction.** SDSs are not the only conversational partners, to whom we adapt our behaviour. We act in this way if we realise that our addressee may have some communicational difficulties, and therefore we emphasise and simplify our speech as soon as we start talking to such special conversational agents [Batliner, Schuller, et al., 2008; Schuller et al., 2017; Siegert and Krüger, 2018]. Instead of the machine in Figure 1.2, there may also be a hard-of-hearing person, a person suffering from psycho- or neurological disorders, or a child having a lack of communicational experience. In all the cases, it is possible to compare the normal manner of speech, that ordinary adults without any impairments causing miscommunication use to address each other, and a specific manner of speech adapted to the special interlocutors. According to this difference, a monitoring system can measure Interaction Quality (IQ) [Spirina et al., 2016] to assess the progress or regress that the special interlocutors are making in maintaining spoken conversations. The obtained results can be used for medical or educational purposes.

### 1.3.2 Joint Analysis of Various Statements of the Addressee Detection Problem

The existing works studied individual special cases of the AD problem separately. Within the present thesis, we jointly analyse the three special cases from Section 1.2 to determine more general, universal patterns that characterise humans’ conversational behaviour when the people change their addressees. We show differences and similarities between the special cases and also improve the AD performance in general as well as for individual problem statements.

### 1.4 Contribution

For the past years, considerable progress has been made towards understanding multiparty H-H conversations [Busso et al., 2007; Akker and Traum, 2009; Gilmartin et al., 2018], and the discovered AD patterns have been adapted to H-H-M conversations [Batliner, Hacker, and Nöth, 2008; Tsai et al., 2015; Ouchi and Tsuboi, 2016]. The main objective of the present thesis is the application and evaluation of novel approaches aimed at improving the performance of existing systems on AD.

It is important to note that AD belongs to paralinguistic problems that are known to be highly language-, corpus-, and even speaker-dependent [Sidorov et al., 2014; Siegert, Shuran, et al., 2018; Egorow et al., 2018]. There exist several automatic experimental systems capable of predicting addresses given various unimodal audio- [Shriberg et al., 2012; Shriberg et al., 2013], text- [H. Lee et al., 2013; S. V. Ravuri and Stolcke, 2014; S. Ravuri and Stolcke, 2015], and video-based [Skantze and Gustafson, 2009] signals. Few studies were dedicated to multimodal H-M AD [Batliner, Hacker, and Nöth, 2008; Tsai et al., 2015]. Nevertheless, each of these systems was tested on a single corpus that was modelling a single SDS and a single statement of the classical H-M AD problem. These limitations complicate the usage of the proposed systems in variable real-live applications. Furthermore, each of the aforementioned studies was confined to one language (usually English). Another essential drawback of these studies is their relatively outdated ML methodology; most of them, except [S. V. Ravuri and Stolcke, 2014; S. Ravuri and Stolcke, 2015] did not use DNNs at all, despite the promising capabilities of such models. Mallidi et al. [2018], Norouzian et al. [2019], and
C.-W. Huang et al. [2019] trained their huge DNN-based models for acoustic H-M AD on vast corpora containing several hundred hours of speech data, although their models were not end-to-end. However, we do not have that much data for training our models.

In this light, the present thesis has the following aims:

- To jointly study various AD corpora which model different problem statements, SDSs, scenarios, domains, and languages (besides English). To determine various corpus-specific and corpus-independent aspects of AD.
- To perform a throughout multimodal analysis of the classical H-M AD problem.
- To propose competitive classifiers, including models based on deep learning in general and end-to-end models in particular, for effective AD.
- To propose a data augmentation method that would facilitate the training of deep models for AD under a lack of training data.

For the first aim, we consider the following corpora: the Voice Assistant Conversation Corpus (VACC), the SmartWeb Video Corpus (SVC), the Restaurant Booking Corpus (RBC), and the HomeBank Child/Adult Addressee Corpus (HB). VACC and SVC model similar problems of classical H-M AD with a desktop SDS and a portable SDS respectively. RBC represents a slightly different problem of complexity-identical H-M AD with a call centre SDS. HB models another problem of H-H AD with a passively listening SDS. The latter corpus contains English speech, while the others were uttered in German. Using the aforementioned corpora, we carry out a cross-corpus study on acoustic and textual AD to find relationships between the classical H-M AD, the complexity-identical H-M AD, and the H-H AD problem.

For the second aim, we analyse speech, text, and a user’s visual focus of attention regarding the classical H-M AD problem of SVC [Batliner, Hacker, and Nöth, 2008] by means of ML approaches. We propose various methods for modelling different components of each modality, estimate their relative contributions to the AD performance, and also perform their fusion. Additionally, we investigate the performance of the proposed models on speech categories of various spontaneity levels.

For the third aim, we propose several classifiers, including some DNN-based models and an end-to-end model, that outperform the existing AD baselines on the considered corpora.

For the fourth aim, we introduce a novel approach to cross-corpus data augmentation, based on the mixup method [H. Zhang et al., 2017], that regularises DNNs and not just improves their single-task performance on a specific corpus, but also enables effective multitask learning on several corpora at once. The corpora being leveraged for the data augmentation may be collected for various (but similar) tasks and even uttered in different languages.

### 1.5 Outline

The present thesis consists of 7 chapters. In this Chapter 1, we introduce several AD problem statements, emphasise the necessity of automatic AD for modern SDSs, and list the key contributions of the thesis. In Chapter 2, we discuss some studies and methods relevant to AD. Chapter 3 is dedicated to the corpora on which we conduct our experiments. In the same chapter, we briefly describe the software and hardware solutions which we employ to carry out the experimental part of our study. In Chapter 4, we introduce several classifiers for speech and text and also conduct a series
of cross-corpus experiments on four different corpora which represent various AD scenarios. As a result, we determine differences and similarities between the corpora in the sense of audio- and text-based AD. Chapter 5 is dedicated to detailed relationships between classical H-M AD and a couple of special cases (H-H AD and complexity-identical H-M AD) in the sense of acoustic and textual AD. In the same chapter, we also introduce a novel approach to cross-corpus data augmentation based on the mixup method. In Chapter 6, we carry out a series of experiments on multimodal H-M AD, jointly using speech, text, and video classification. Finally, we make concluding remarks and propose possible directions for future research in Chapter 7.
Background and Related Work

The use of ML techniques is the core concept of modern computational paralinguistics. In the present chapter, we discuss several studies and methods which are relevant to AD or to similar problems of computational paralinguistics. All studies mentioned in the related work (Section 2.1) follow a similar architectural concept. We can visualise it as the following pipeline:

1. Feature extraction;
2. Feature selection and feature transformation (optional);
3. Classification;
4. Data fusion (optional).

The individual stages of this pipeline are described in Section 2.2 – 2.5.

The present chapter is organised as follows. First, we analyse related work in Section 2.1. Section 2.2 is dedicated to various data modalities and feature sets which are relevant to AD. In Section 2.3, we describe several universal approaches to dimensionality reduction. In Section 2.4, we discuss various ML models which can be employed as classifiers for AD. Finally, we consider several fusion techniques in Section 2.5. The content of the present chapter is summarised in Section 2.6.

2.1 Related Work

All studies discussed in this section represent H-M AD as an utterance classification problem. Many existing works on H-M AD, e.g., [Batliner et al., 2006; Shriberg et al., 2012; Shriberg et al., 2013; Mallidi et al., 2018; C.-W. Huang et al., 2019], study speech jointly with text. It is reasonable since both acoustic and textual features can be extracted from the audio modality. Few studies, e.g., [Batliner, Hacker, and Nöth, 2008; Tsai et al., 2015], also consider additional modalities, such as video.

2.1.1 "Smart Web" and Other Multimodal Studies

Batliner et al. [2006] introduced three corpora for H-M AD. The first corpus was collected using the SmartKom SDS. This system was simulated by the Wizard-of-Oz method (WOZ). By means of the SDS, a user could ask queries on specific points of interest, e.g., hotels, restaurants, or cinemas. The SDS supported unconstrained spoken input, synthesised spoken output, and also graphical output.
Since the user was alone, human-directed utterances mostly consisted of talking to oneself and were sparse. The second corpus was collected using the SmartWeb SDS, which is the next version of SmartKom. The SmartWeb data, also known as the SmartWeb Video Corpus or SVC, has some differences and improvements compared to SmartKom. First, SVC models a realistic, spontaneous H-H-M conversation between a user, a companion, and an SDS. Second, SVC includes the visual modality. Third, SVC has a different domain; the SmartWeb SDS operated as a tourist guide for the Football World Cup 2006. The third corpus is quite similar to SVC. Its main difference from SVC is the humans’ acted behaviour; the speakers got instructions on what text to pronounce and how to pronounce it for On- and Off-Talk.

Another study on multimodal H-M AD was conducted by Tsai et al. [2015]. They used a corpus from a multiparty conversational setup described by Bohus and Horvitz [2010]. The scenario involved groups of two or three people playing a trivial question game with an SDS. The system was represented as a talking face displayed on a large monitor. The SDS asked the group a question, gave the participants some time to discuss possible answers, listened to the final answer, and then told them if their answer was correct.

Related Papers

Batliner, Hacker, and Nöth [2008] extended the paper [Batliner et al., 2006] by introducing an audio-visual approach to classical H-M AD. However, both papers share the same methodology for speech- and text-based H-M AD. Batliner, Hacker, and Nöth [2008] computed various groups of acoustic-prosodic features (duration, energy, F0, length of pauses, jitter, shimmer, and rate of speech) within the current word and also in the context of up to two words before and after the current word for which an addressee label needed to be predicted. The authors called this approach "prosodic n-gram". Assuming that each user’s speech was known to the system beforehand, the authors tried speaker normalisation, bringing each acoustic feature to zero mean and unit variance. Similarly to the acoustic features, Batliner, Hacker, and Nöth [2008] computed Part-of-Speech (POS) n-grams considering the POS tag of the current word and also in the context of up to two words before and after the current word for which an addressee label needed to be predicted. After that, the authors concatenated the acoustic and the POS features and used a linear discriminant classifier to predict an addressee for each word. Finally, Batliner, Hacker, and Nöth [2008] introduced an audio-visual classifier that consisted of the following three models. The first model worked only with the acoustic features, the second one dealt only with the POS features. Both models were based on linear discriminant classifiers. The third model used a frontal face detector based on the Viola-Jones algorithm [Viola and Jones, 2004]. It detected On-View frames (in which a user looks at the camera placed above the system display). After that, various characteristics were extracted from the obtained sequence of frame-level On-/Off-View predictions. Similarly, some metafeatures were extracted from the sequences of word-level On-/Off-Talk predictions obtained from the acoustic and POS models. All three groups of utterance-level metafeatures were concatenated and fed to another linear discriminant classifier that returned utterance-level addressee predictions.

Batliner, Hacker, and Nöth [2008] used Unweighted Average Recall (UAR) to evaluate the proposed classifiers. We are interested in the two-class categorisation problem (On- vs. Off-Talk) only. The acoustic features outperformed the POS features, and the speaker normalisation of the acoustic features further improved the results for all three corpora. The fusion of the acoustic and POS
features also improved the performance of individual modalities and provided UAR values of 0.741 and 0.681 for the SmartKom corpus and SVC respectively. The speaker-normalised acoustic features reached a UAR of 0.926 on the SmartWeb acted corpus; POS features do not make much sense in this case, since the utterances of the latter corpus were scripted and then pronounced word-for-word. On the one hand, cross-corpus experiments revealed the SmartWeb acted data to be significantly easier to classify, even without using this corpus for training, compared to the other two corpora. On the other hand, the systems employing the SmartWeb acted corpus alone for training demonstrated adequate performance values on the other two corpora. The UAR values given in the present section above were obtained for H-M AD at the word level. The UAR values given in this section below correspond to H-M AD at the utterance level. The individual models of the audio-visual classifier demonstrated UAR values of 0.766, 0.760, and 0.705 for speaker-normalised prosody, POS, and video respectively. The fusion of prosody and POS improved the classification performance ($UAR = 0.808$), while adding the visual modality gave a further improvement ($UAR = 0.845$).

Tsai et al. [2015] extracted a large variety of features from their multimodal corpus. The features included five modalities: acoustic, visual, system, beamforming, and ASR. The acoustic attributes consisted of energy, energy change, and temporal energy contour features. They are similar to the features proposed by Shriberg et al. [2012] and Shriberg et al. [2013]. The visual attributes comprised amount-of-movement, face orientation, and physical distance features. The system attributes consisted of various indicators of the system state, including the dialogue state. The beamforming attributes described the distribution of beam values, i.e., the direction of audio waves coming from the users. The ASR attributes comprised $n$-gram-based features, which are similar to those proposed by Shriberg et al. [2013], and features describing various properties of the ASR hypotheses, which are similar to those proposed by Shriberg et al. [2012]. More information on the features employed by Tsai et al. [2015] can be found in Section 2.2.

Tsai et al. [2015] experimented with four different types of classifiers: logistic regression, decision tree, random forest, and Adaboost [Freund et al., 1999] with tree stumps. All decision trees employed in their study were regression trees. The authors also performed early fusion using an approach similar to [Pérez-Rosas et al., 2013].

Tsai et al. [2015] used Equal Error Rate (EER) and Detection Error Tradeoff (DET) to evaluate the proposed models. Adaboost outperformed the other classifiers for all five modalities. Therefore, we omit the results of the other classifiers. The authors ranged the modalities in accordance with their relative contribution to the total classification performance and obtained the following sequence: acoustic (most important), ASR, system, visual, beamforming (least important). The acoustic modality alone provided an EER of 0.139, while the fusion of all five modalities significantly improved the latter result ($EER = 0.098$).

Summary

According to Batliner, Hacker, and Nöth [2008] and Tsai et al. [2015], it is worth analysing other modalities in addition to speech and text for more effective H-M AD. However, according to the results obtained, speech and text make the most significant contribution to the total AD performance of the multimodal approaches. The largest drawback of the studies [Batliner, Hacker, and Nöth, 2008; Tsai et al., 2015] is their outdated ML methodology. In particular, they used sophisticated
Background and Related Work

preprocessing methods to extract highly abstract features that would work well with very simple classifiers.

We believe that employing lower-level features, e.g., distributed word representations for text or sequences of frame-based acoustic descriptors for audio, can keep a larger amount of useful information for AD. Modern ML methods based on DNNs can unlock the potential of low-level features and improve the AD performance. We address this issue in the present thesis.

2.1.2 "Conversational Browser" Studies

Shriberg et al. [2013] employed an extended version of the data set they previously introduced in [Shriberg et al., 2012]. The data was collected using an SDS called "Conversational Browser" (CB) that was capable of understanding a small set of short, simple spoken utterances. The system was showing a browser on a large TV screen and two users were asked to employ a natural language for some cooperative information-seeking tasks. This setup modelled classical H-M AD. More information about the system and its spoken language understanding approach can be found in [Heck et al., 2013].

Related Papers

Compared to [Shriberg et al., 2012], Shriberg et al. [2013] investigated a larger set of acoustic features and conducted an out-of-domain training. However, both papers are quite similar in general. Shriberg et al. [2013] extracted various textual and acoustic-prosodic features to solve the classical H-M AD problem. The textual features comprised lexical n-grams of words including start/end-of-utterance tags. The acoustic-prosodic features consisted of energy contour features, voice quality and spectral tilt features, and delta energy features. The energy contour features used 10-ms-frame c0 and c1 output from standard MFCCs mean-normalised over the utterance. After that, a Discrete Cosine Transform (DCT) was taken over a 200-ms sliding window with a 50% shift. The resulting features comprised the first 5 and 2 bases from the DCT over each window of c0 and c1 respectively. The voice quality and spectral tilt features were extracted for voiced regions only. These features included H2-H1, F1-H1, F2-H1 (all three using lower-order harmonics and formants), the spectral slope per frame (computed as the slope of a linear least squares fit to the log spectrum), and the difference between the maximum of the log power spectrum and the maximum in the 2kHz-3kHz range. The delta energy features represented the difference in log energy between frames centred around the voicing onset/offset.

For text classification, Shriberg et al. [2013] computed a log likelihood ratio of the two addressee classes from lexical n-grams by modelling each class with a maximum entropy language model (LM). The energy contour and tilt features employed Gaussian Mixture Models (GMMs) to compute a log likelihood ratio. Training feature vectors for each class were pooled and a GMM with full covariances was trained. Utterance-level features (both real-valued and binary) were modelled by the adaptive boosting algorithm [Schapire and Singer, 2000]. For features that generated multiple vectors per utterance, Shriberg et al. [2013] first computed statistics (max, min, mean, standard deviation), both for the entire utterance and over regions of contiguous feature vectors, and used the obtained functionals for boosting. Linear Logistic Regression (LLR) was applied to one or more decision scores obtained with any of the aforementioned classifiers.
EER and DET were used to evaluate the proposed models. The system leveraging all the prosodic features reached an EER of 0.125. This result is significantly better than the performance of the ASR-based \(n\)-grams (EER = 0.270) but worse than the performance of the \(n\)-grams based on the manual transcripts (EER = 0.104). The best performing system using all the lexical and prosodic features provided EER values of 0.109 and 0.075 for the ASR output and manual transcripts respectively. In their online processing experiments, Shriberg et al. [2013] determined the first two seconds of each utterance to make the largest contribution to the total AD performance for both acoustic and textual modality. In their out-of-domain training experiments, Shriberg et al. [2013] employed several corpora (completely different from the target corpus) containing only machine-directed or only human-directed utterances to train the proposed acoustic models for the respective addressee classes. The most effective of these classifiers demonstrated an adequate EER of 0.200 on the target data of CB, although the models did not use this corpus for training.

H. Lee et al. [2013] and S. V. Ravuri and Stolcke [2014] used the same idea of out-of-domain training and the same in-domain data of CB as Shriberg et al. [2013] did. However, the studies [H. Lee et al., 2013; S. V. Ravuri and Stolcke, 2014] were focused on the textual modality.

H. Lee et al. [2013] computed a log likelihood ratio of the two addressee classes from lexical \(n\)-grams by modelling each class with a standard trigram backoff LM. Witten-Bell discounting was applied for smoothing [Witten and Bell, 1991]. For combining various training data sources, H. Lee et al. [2013] used language model adaptation by interpolation [Bellegarda, 2004]. H. Lee et al. [2013] also preserved \(N\) most frequent words in the training and test data, while other words were replaced by their POS tags to resolve the lack of statistical information for such words. The \(N\) hyperparameter was tuned empirically. After that, the authors applied the aforementioned lexical model to the augmented data as before.

The classifier that leveraged both in- and out-of-domain training data demonstrated EER values of 0.266 and 0.130, while the classifier that was trained only on the in-domain data reached worse values of 0.311 and 0.173 for the ASR output and manual transcripts respectively. The classifier that used only the out-of-domain training data provided adequate EER values of 0.278 and 0.142 for the ASR output and manual transcripts respectively. The best performing configuration of the classifier using the augmented data (with POS tags, \(N = 300\)) was trained on both in- and out-of-domain corpora. This configuration outperformed the initial model (without POS tags) and reached EER values of 0.261 and 0.125 for the ASR output and manual transcripts respectively.

S. V. Ravuri and Stolcke [2014] extended the study [H. Lee et al., 2013] by introducing feedforward-ANN-based LMs to solve the classical H-M AD problem on the data of CB. However, S. V. Ravuri and Stolcke [2014] obtained only minor classification performance improvements over the traditional LMs.

In their follow-up study, S. Ravuri and Stolcke [2015] investigated applications of RNNs and LSTM neural networks on the same problem. Both RNN and LSTM models used 200-dimensional word embeddings trained on the in-domain data of CB. The word embeddings were extracted from \(|V|\)-dimensional one-hot vectors, where one element equals 1, the others equal 0, and \(|V|\) is the number of all unique words, i.e., vocabulary size, in the training data. In addition to the traditional word embeddings, S. Ravuri and Stolcke [2015] extracted embeddings from word hashes to handle out-of-vocabulary words as proposed in [P.-S. Huang et al., 2013]. S. Ravuri and Stolcke [2015] applied LLR to one or more decision scores obtained with any of the aforementioned classifiers.
S. Ravuri and Stolcke [2015] achieved the highest classification performance by fusing the baseline maximum entropy LM based on word n-grams with the RNN LM based on word hashes. This classifier reached an EER of 0.231 that is significantly better than the performance of the baseline model \((EER = 0.270)\).

Summary

The papers [Shriberg et al., 2012; Shriberg et al., 2013; H. Lee et al., 2013; S. V. Ravuri and Stolcke, 2014; S. Ravuri and Stolcke, 2015] considered in the present section above have significant drawbacks. Their most critical issues are connected with the data of CB. First, it is a single corpus that the authors employed for testing the proposed classifiers. Although S. Ravuri and Stolcke [2015] used another corpus in addition to CB for testing, this additional corpus had been collected for a classification problem that differed from AD. Second, it may be difficult to validate the proposed systems on other corpora, given the simplicity of the spoken language supported by the CB SDS. Testing the prosed systems on several data sets which have different scenarios and characteristics is important to obtain a credible assessment of their classification performance. Third, the ML methodology used by the authors looks outdated nowadays. Shriberg et al. [2012], Shriberg et al. [2013], and H. Lee et al. [2013] did not employ any ANN-based models at all. Although S. V. Ravuri and Stolcke [2014] used a feedforward-ANN-based LM, it provided only a minor improvement over a traditional maximum entropy LM, since such an ANN is not meant to handle text as time-continuous input. In their follow-up study, S. Ravuri and Stolcke [2015] replaced the feedforward-ANN-based LM by an RNN-based LM and obtained a significant performance improvement over the former model, since the latter model was designed to handle text as time-continuous input. However, S. Ravuri and Stolcke [2015] employed an outdated approach to training word embeddings without using out-of-domain textual data. There exist modern methods, e.g., Word2Vec or GloVe, the resulting word vectors of which can be trained on out-of-domain textual data that makes them domain independent and appropriate for a large variety of tasks.

In the present thesis, we address the aforementioned issues by employing several corpora which model different AD scenarios. These corpora contain realistic, spontaneous conversations which are used for both training and testing our models. We also propose modern, highly effective approaches to audio- and text-based AD that include DNNs and out-of-domain training.

2.1.3 Studies of Deep Neural Networks

Mallidi et al. [2018] used a huge corpus (350 000 and 50 000 utterances, 250 hours and 30 hours of speech in the training and the test set respectively) that contained real recordings of natural interactions between humans and voice-controlled far-field devices. The corpus was employed for H-M AD.

Mallidi et al. [2018] proposed a large DNN having a huge number of trainable parameters. This model included two LSTM neural networks. The first LSTM operated with log filter-bank energies which were obtained using a sliding window of 25 milliseconds and a 10 milliseconds shift. The model was trained to predict frame-level addressee targets which were obtained by repeating the utterance label. After the training, the pre-softmax output of the last frame of the input utterance was used as its representation. Similarly to the first model, the second LSTM was trained on a sequence
of pretrained character embeddings [Pennington et al., 2014] to predict character-level addressee targets. After the training, the network output of the last character of the recognised input utterance was used as its representation. A fully connected ANN was then trained to combine some ASR decoder features, e.g., recognition confidence, and the obtained acoustic and textual embeddings.

Mallidi et al. [2018] used EER to evaluate the proposed classifiers. The textual, acoustic, and ASR decoder features reached EER values of 0.201, 0.109, and 0.093 respectively. The fusion of all features resulted in a significantly better EER of 0.052.

In their follow-up study, C.-W. Huang et al. [2019] determined that a similar model with a separate output branch for each dialogue type outperformed the model with a shared output branch from [Mallidi et al., 2018]. Inspired by [Norouzian et al., 2019], C.-W. Huang et al. [2019] also managed to improve the EER of their system by applying an attention pooling mechanism to training acoustic embeddings.

Summary

Given the huge available amount of training data, Mallidi et al. [2018], Norouzian et al. [2019], and C.-W. Huang et al. [2019] developed large DNNs having a lot of trainable parameters, although none of the proposed models were end-to-end. For example, the best performing configuration of the acoustic model proposed by Mallidi et al. [2018] had around 16 million of learnable parameters. However, we do not have that much training data at our disposal. To address this issue, we train our classifiers on several data sources and also introduce a novel approach to data augmentation based on the mixup method [H. Zhang et al., 2017]. This approach is particularly useful for combining different data sources in a smooth manner and also for regularising complex DNN-based models. In addition, we propose an end-to-end DNN for AD.

2.2 Feature Extraction

Let us consider various multimodal aspects of an H-M AD problem presented by Tsai et al. [2015]. Their work is one of the most thorough studies on multimodal H-M AD to our knowledge, and therefore it is worth to be given as a vivid example. First, we are supposed to define representative modalities and relevant features that need to be extracted from them. In order to do this, let us analyse the data that Tsai et al. [2015] had at their disposal and the conditions under which it was collected. The authors decided to use a multimodal corpus from a multiparty conversation setup designed by Bohus and Horvitz [2010]. Its scenario involves groups of several humans playing a trivial question game with an artificial conversational agent. The people interact with the machine through an avatar that is represented as a talking face displayed on a 19-inch computer monitor. The avatar supports various head poses and facial expressions. It engages with the participants through dialogue and face movement. The agent asks a group a question, confirms the result of the discussion between the participants, and then tells them if their answer is correct. The participants within each group know each other. The corpus design is aimed at studying computational models of multiparty turn-taking and encourages realistic, fluid interactions. This experimental setup depicted in Figure 2.1 is a real-life interpretation of the interaction scheme from Figure 1.2.
The audio data is recorded with a linear microphone array placed above the computer monitor. The microphone array can capture the users’ voices together with their beamforming. The audio is further processed using the integrated Windows ASR. The video data is collected with a wide-angle camera located above the microphone array. The system performs real-time video processing to detect each participant’s face in each frame. After that, the detections across frames are associated using a proximity-based algorithm [C. Zhang and Rui, 2006]. Each detected face is subjected to three-dimensional head orientation estimation with a Bayesian pose-tracking algorithm [Q. Wang et al., 2006]. Both the microphone array and the camera tracking a user’s focus of visual attention can be seen in Figure 2.1 directly above the computer monitor. Besides audio and video, the system logs various aspects of the interaction, e.g., dialogue states, the conversation history, the quantity of the participants in the current group, the computer agent’s responses, and who the agent is looking at. More details on the setup can be found in the works [Bohus and Horvitz, 2009; Bohus and Horvitz, 2010]. The recorded data was automatically segmented into utterances with a voice activity detector. The resulting utterances were manually annotated with speech, speaker, and addressee information by human experts.

As a result, the collected data includes the following modalities:

- acoustic;
- beamforming;
- ASR;
- visual;
- system.

Let us consider each of them in detail.

2.2.1 Acoustic Features

Speech is the basic modality for any SDS, and therefore audio-based AD is considered to be the most flexible solution that can be applied to systems operating in different domains [Shriberg et al., 2013]. In this section, we discuss various representations of acoustic information for AD.
Energy Features

Tsai et al. [2015] extract the three families of acoustic features listed below. The first family contains energy features, i.e., various functionals (min, max, average, and others) computed over frame-level energies throughout each utterance as well as on various time intervals of up to three seconds before and after the utterance. Examples of the acoustic features belonging to the first family are as follows:

- average energy during the first third of each utterance;
- maximum frame-level energy throughout each utterance;
- average energy during the one-second interval preceding each utterance.

The intuition behind the features of the first family is based on the observation that humans tend to increase the loudness of their speech as soon as they start talking to an SDS [Tsai et al., 2015].

Energy Change Features

The second family of acoustic features comprises energy change features reflecting the difference in energy between two neighbouring time intervals within each utterance and intervals of up to three seconds before and after the utterance [Tsai et al., 2015]. Examples of the acoustic features belonging to the second family are as follows:

- difference between the average energy during each utterance and the average energy during the two-second interval after the utterance;
- difference between the maximum frame-level energy during the first third of each utterance and the maximum frame energy during the one second before the utterance.

The idea behind the features of the second family is that people tend to raise their voices while pronouncing a machine-directed utterance and make longer pauses after talking to a computer than after talking to another human since they have to wait for the machine’s response due to considerable precessing time [Tsai et al., 2015].

Temporal Energy Contour Features

The third family of acoustic features describes the temporal shape of the speech energy contour, as proposed by Shriberg et al. [2012]. Zeroth- and first-order Mel-Frequency Cepstral Coefficients (MFCCs) are calculated at intervals of 10 milliseconds, and the contours of these values are characterised by computing a Discrete Cosine Transform (DCT) over temporal windows of 200 milliseconds. MFCC coefficients are usually used for speech recognition [Zhen et al., 2000]. The intuition behind the features belonging to the third family is based on the following observation: Shriberg et al. [2012] noted that users employ a more regular rhythmic up-and-down energy pattern for machine-directed utterances than for human-directed ones, i.e, computer-addressed speech sounds more "sing-songy". In contrast to the other two families of acoustic features, which are extracted at the utterance level, the energy contour features are frame-level attributes that require a frame-to-utterance-level transformation for the fusion with the other feature families. Two Gaussian Mixture Models (GMMs) are trained for this purpose (one for each target class). A single utterance-level energy contour feature value is obtained computing a log likelihood ratio of these two binary classifiers. Alongside with this feature, its analogue normalised by utterance length, i.e., by the number of frames, is used.
**Proso dic N-Grams**

Batliner, Hacker, and Nöth [2008] expect addressee patterns to appear between the word and the utterance level, e.g., at the levels of collocations and phrases. The authors apply a so-called "prosodic n-gram" to speech analysis. This approach is an adaptation of the classical n-gram method that is widely used for text analysis. Batliner, Hacker, and Nöth [2008] solve an H-M AD problem and determine addressees at the word level. First, they align speech with recognised text in order to find starting and ending time points for each word. After that, they compute various acoustic-prosodic features within the current word and also in the context of up to two words before and after the current word for which an addressee label needs to be predicted. The following groups of acoustic attributes are used:

- duration;
- energy;
- F0;
- lengths of pauses;
- jitter;
- shimmer;
- rate of speech.

Duration, energy, F0, and a rate-of-speech feature are also estimated globally for a context window of 15 words or fewer if the current utterance is shorter. This feature design results in 100 acoustic attributes in total. More details on the feature design can be found in [Batliner et al., 2000; Batliner et al., 2003]. The fact that the prosodic n-gram relies on accurate ASR (Batliner, Hacker, and Nöth [2008] used manual transcripts, i.e., the Word Error Rate (WER) was equal to zero) is the most significant disadvantage of this approach. Another drawback of this method is connected with the previous assumption that addressee patterns arise beyond the word level, and therefore the feature design does not allow us to have a look at intra-word levels, e.g., syllables or phonemes. Meanwhile, the intra-word levels may contain additional acoustic information about addressees. One more disadvantage of the proposed approach is the complexity of the feature design that may potentially cause information losses. The prosodic n-gram represents time continuous acoustic data as a single vector, while there exist more natural representations for such data, e.g., Low-Level Descriptor (LLD) contours [Eyben, 2015]. The prosodic n-gram has a limited context coverage and captures only word-level features, and it is therefore unclear how this highly abstract single-vector representation performs on other corpora designed for various AD scenarios besides the classical H-M AD one investigated by Batliner, Hacker, and Nöth [2008]. The authors had to confine to the single-vector representation since they used it in combination with a simple Linear Discriminant Classifier (LDC) [Klecka, 1988]. This model is capable of performing both feature transformation and classification at once, though it is not meant for handling time continuous input. However, some other models, e.g., Recurrent Neural Networks (RNNs) and their Long Short-Term Memory (LSTM) versions [S. Hochreiter and Schmidhuber, 1997], were specially designed for handling low-level time continuous features containing important temporal information. As any DNN, the RNN-based models can also perform feature selection and transformation for a specific classification problem.
2.2 Feature Extraction

IS13_ComParE Features

AD was a sub-challenge topic of the Interspeech 2017 Computational Paralinguistics Challenge (ComParE) [Schuller et al., 2017]. The challenge organisers used a large acoustic feature set in their baseline models for solving the problem of H-H AD. This attribute set was introduced within the Interspeech 2013 ComParE challenge [Schuller et al., 2013] and was also used in all the subsequent challenges. Let us denote this feature set just as IS13_ComParE for simplicity. Besides H-H AD [Schuller et al., 2017], the IS13_ComParE features were shown to be effective for a wide range of paralinguistic problems, e.g., emotion recognition [Schuller et al., 2013], native speech detection [Schuller et al., 2015], neurological pathology detection [Schuller et al., 2013; Schuller et al., 2015], detection of persons having colds [Schuller et al., 2017], snoring classification [Schuller et al., 2017], and many others. The IS13_ComParE feature set consists of 6373 attributes resulting from the computation of various functionals over 130 LLD contours: energy, spectral, MFCC, and voicing related LLDs as well as a few LLDs including logarithmic Harmonic-to-Noise Ratio (HNR), spectral harmonicity, and psychoacoustic spectral sharpness. Detailed descriptions of the OpenSMILE feature extractor and its default IS13_ComParE feature configuration can be found in [Eyben et al., 2010; Eyben et al., 2013] and [Eyben, 2015] respectively. The IS13_ComParE LLDs are extracted at the frame level with a frame length of 60 milliseconds and an overlap of 50 milliseconds by default, while their functionals are computed for the entire audio file. The IS13_ComParE feature set includes the following 65 LLDs alongside with their deltas:

4 energy-related LLDs:
- audspec_lengthL1norm_sma (prosodic) – sum of auditory spectrum (loudness);
- audspecRasta_lengthL1norm_sma (prosodic) – sum of auditory spectrum after Relative Spectral Transform (RASTA-style) filtering (modulation loudness);
- pcm_RMSenergy_sma (prosodic) – Root Mean Square (RMS) frame-based energy of time signal;
- pcm_zcr_sma (prosodic) – frame-based zero crossing rate of time signal;

55 spectrum-related LLDs:
- audSpec_Rfilt_sma[0] & ... & ..._sma[25] (spectral) – RASTA-style auditory spectrum, bands 1-26 (0-8kHz);
- mfcc_sma[1] & ... & ..._sma[14] (cepstral) – MFCC coefficients 1-14 without the zeroth one. This coefficient was removed due to its redundancy with RMS energy. The MFCCs are computed from a power-mel spectrum (26 bands, triangular filter, range of 20 – 8000 Hz);
- pcm_fftMag_fband250-650_sma & _...1000-4000_sma (spectral) – magnitude of low (250-650 Hz) and high (1000-4000 Hz) frequency bands of time signal after Fast Fourier Transform (FFT) (spectral energy);
- pcm_fftMag_spectralRollOff25.0_sma & _...50.0_sma & _...75.0_sma & _...90.0_sma (spectral) – 25, 50, 75, and 90% spectral roll-off points;

1 The feature notations are given as specified in the IS13_ComParE.conf file.
2 Delta (denoted as an LLD name with the appended _de suffix) means a first-order delta coefficient (differential) computed for each LLD.
3 The _sma suffix means that the LLD contour was smoothed by a moving average filter with a window length of three.
4 Feature category (prosodic, spectral, cepstral, or voice quality).
24 2 Background and Related Work

- **pcm_fftMag_spectralFlux_sma** (spectral) – normalised spectral difference between two successive frames (spectral flux);
- **pcm_fftMag_spectralCentroid_sma** (spectral) – spectral linear centroid;
- **pcm_fftMag_spectralEntropy_sma** (spectral) – spectral entropy;
- **pcm_fftMag_spectralVariance_sma** (spectral) – spectral variance;
- **pcm_fftMag_spectralSkewness_sma** (spectral) – spectral skewness;
- **pcm_fftMag_spectralKurtosis_sma** (spectral) – spectral kurtosis;
- **pcm_fftMag_spectralSlope_sma** (spectral) – spectral slope;
- **pcm_fftMag_psySharpness_sma** (spectral) – psychoacoustic spectral sharpness;
- **pcm_fftMag_spectralHarmonicity_sma** (spectral) – spectral harmonicity;

6 voicing-related LLDs:
- **F0final_sma** (prosodic) – fundamental frequency contour;
- **voicingFinalUnclipped_sma** (voice quality) – voicing probability of the final fundamental frequency candidate. Unclipped means that it is not set to zero when it falls below the voicing threshold;
- **jitterLocal_sma** (voice quality) – local frame-to-frame Jitter (pitch period length deviations);
- **jitterDDP_sma** (voice quality) – differential frame-to-frame Jitter (the "Jitter of the Jitter");
- **shimmerLocal_sma** (voice quality) – local frame-to-frame Shimmer (amplitude deviations between pitch periods);
- **logHNR_sma** (voice quality) – logarithmic HNR.

These 65 LLDs alongside with their deltas provide 130 frame-level acoustic contours in total. The following *functionals* are computed over these contours:

- Extreme values and positions;
- Means (arithmetic, quadratic, geometric);
- Moments (standard deviation, variance, kurtosis, skewness);
- Percentiles and percentile ranges;
- Regression (linear and quadratic approximation, regression error);
- Centroid;
- Peaks;
- Segments;
- Sample values;
- Times/durations;
- Onsets/offsets;
- DCT;
- Zero-crossings;
- Linear Predictive Coding (LPC) coefficients and gain.

The functional extraction from the IS13_ComParE LLDs results in 6 373 utterance-level features in total.

It is important to note that the IS13_ComParE attributes (LLDs, deltas, and functionals) contain all the families of acoustic features investigated by Tsai et al. [2015]. However, the IS13_ComParE attribute set is much larger than the feature set examined in the latter study, and therefore the
IS13_ComParE features allow us to capture all the variety of continuous spontaneous speech. A certain degree of redundancy of the IS13_ComParE feature set with respect to classical acoustic H-M AD will come in handy since we do not confine to a single problem statement as Tsai et al. [2015] did. Besides classical H-M AD, we also analyse other AD problems: complexity-identical H-M AD and H-H AD. Furthermore, the corpora at our disposal were collected in different SDS scenarios under various acoustic conditions, while the data set from [Tsai et al., 2015] was recorded within a single SDS scenario in a nearly ideal acoustic environment of a lab. The IS13_ComParE LLDs applied in combination with a recurrent model, such as RNN and LSTM, reproduce the concept of the prosodic $n$-grams with essential improvements: such a sequential model does not require ASR, and it is no longer limited with word-level analysis (it is only limited with the resolution of the LLD computation). Furthermore, it processes time continuous information in a more natural way and can cover much longer acoustic context compared to the prosodic $n$-grams proposed by Batliner, Hacker, and Nöth [2008]. Taking all the mentioned peculiarities into account, we make our choice in favour of the IS13_ComParE feature set.

**Raw Audio Data**

Any data preprocessing leads to information losses. Ideally, we need to use acoustic information as it is in audio files, i.e., as raw waveforms. DNN-based end-to-end systems have had huge success within the visual processing community and recently became popular in the sound processing community as well. Such models allow us to analyse raw data; they are capable of performing end-to-end audio processing for various problems, e.g., emotion recognition [Trigeorgis et al., 2016], speaker verification [Heigold et al., 2016], speech recognition [Oord et al., 2016], acoustic scene/object classification [Aytar et al., 2016], and others [Smaragdis, 2016]. On the one hand, the main advantage of DNNs and particularly end-to-end models over classical ones is their ability to learn the optimal highly abstract feature representations jointly with the optimal model parameters for a given task in a single optimisation problem. This ability reduces the chance to choose a sub-optimal feature set manually and also eases the requirements for input feature design. In case of end-to-end models, the input feature design becomes completely unnecessary. On the other hand, such models have essentially more complicated architectures that require an accurate choice of their hyperparameters and a large amount of training data. There have been few attempts to apply end-to-end models to acoustic AD, e.g., Schuller et al. [2017] implemented an end-to-end Convolutional Recurrent Neural Network (CRNN) for acoustic H-H AD. However, the latter model was outperformed by simpler classifiers which were not end-to-end.

**2.2.2 Beamforming Features**

Similarly to the visual modality, beamforming features represent a user’s focus of attention in the geometric sense. They describe the distribution of beam values or, in other words, the intensity of incoming audio signals along various directions. This distribution can be measured with a microphone array like that one used by Tsai et al. [2015]. It is sufficient to place a linear microphone array in front of the speakers in the horizontal plane orthogonally to the imaginary line connecting the system with the middle of the speakers’ group as depicted in Figure 2.1. A flat beam distribution indicates a spoken activity among several interlocutors that is likely to be a sign of human-directed
speech, while a beam distribution with a noticeable peak is an evidence of a single person talking. Furthermore, the shape of the beam distribution allows us to predict the direction in which this person is speaking. The following features can be extracted from the beam distribution:

- variance of the angles of beam values;
- range of the angles of beam values (i.e., the difference between the maximum and the minimum angle);
- fraction of beam values falling within a certain angle range.

These measurements are computed throughout each utterance as well as on various time intervals of up to three seconds before and after the utterance. Tsai et al. [2015] show that using the beamforming features in combination with others provides from little to no benefits for AD compared to the classification without the beamforming information. Furthermore, using the beamforming features alone is not enough for reliable AD.

2.2.3 ASR Features

An accurate ASR provides us with reliable textual information extracted from users’ speech. The problem of AD can be reduced to text classification [Batliner, Hacker, and Nöth, 2008], and therefore we adapt the existing methodology on this topic to text-based AD. This section is dedicated to various numeric representations of text which are relevant to AD. These representations can be divided into the following two categories:

- **Single-vector text representations.** The methods for obtaining these representations are known as Text-to-Vector (T2V) techniques, and the resulting text representations that they produce are called text embeddings. Such a method builds a vector space model for text preprocessing so that the output of this model can be used as the input of an ML algorithm for classification [Sebastiani, 2002]. The main advantage of the T2V methods considered in the present study is their ability to perform well on relatively small amounts of training data. Most classical T2V methods are based on a single pipeline that consists of the following stages:
  1. Tokenisation;
  2. Filtering (optional);
  3. Term extraction (vocabulary forming);
  4. Term weighting;
  5. Dimensionality reduction (optional).

This pipeline returns a single-vector numeric representation for each input piece of text. We will call such a piece of text an utterance or a document in future. We can fine-tune this pipeline to a particular text classification task by enabling different operators at each preprocessing stage. The operators and their properties are described in this section below.

- **Distributed word representations.** Unlike the aforementioned single-vector text representations, distributed word representations are built for single words and not for the entire document. The methods for obtaining these representations are known as Word-to-Vector (W2V) techniques, and the resulting word representations that they produce are called word embeddings. As a rule, W2V methods are unsupervised, i.e., they utilise the statistical information about word co-occurrences in large amounts of unlabelled training data. Any W2V technique
possesses two important properties [Pennington et al., 2014]. The first one is the ability to provide an effective method for measuring the linguistic or semantic similarity between different words. This similarity is usually calculated as the Euclidean distance or cosine similarity between the corresponding words, i.e., the more two words are similar in the semantic sense, the closer they lie to each other in the word vector space. For example, "frog" lies closer to "amphibian" than to "mammal". However, a single distance value is not enough to capture all the relations between two words. The second property is the ability to form linear substructures in the word vector space. In other words, the vector difference operation makes sense in this space: the differences "man"-"woman", "sir"-"madam", and "king"-"queen" are approximately equal since they all capture the relation between a male and a female. More detailed descriptions of W2V methods are given in this section below.

Single-Vector Text Representations

Single-vector text representations using lexical $n$-grams were employed for text-based AD by H. Lee et al. [2013]. The text preprocessing pipeline for obtaining such representations consists of the stages listed below.

**Tokenisation.** At this preprocessing stage, all the sentences of a document are converted into a sequence of words (tokens) separated with spaces. All auxiliary characters, e.g., punctuation marks and special symbols, are deleted, all capital letters are brought to lowercase, etc. [Sergienko, 2016].

**Filtering (optional).** Various linguistic filters are applied at this preprocessing stage. Given the specificity of AD, it makes sense to use one of the following filters:

- **Stop-word filtering.** This filter is an example of a linguistic dimensionality reduction method: it removes pronouns, prepositions, articles, interjections, and other words that usually have no importance for text classification since they do not contain any semantic information related to the document category [Fox, 1989]. As a result, this method significantly reduces the vocabulary size, i.e., the dimensionality of a text classification problem. Stop-word lists are obviously language-dependent and need to be created manually for each language. Examples of stop-word lists can be found in [Blanchard, 2007; Dragut et al., 2009]. An example of stop-word filtering is given in Formula 2.1 – 2.3 and 2.11 – 2.15.

- **Stemming.** This filter is another example of a linguistic approach to dimensionality reduction: it replaces related word forms by their canonical form that is called root, stem, or lemma. The idea behind stemming is to reduce the vocabulary size by eliminating the morphological diversity provided by suffixes and prefixes as shown by Porter [1980]. There exist several approaches to stemming, e.g., lookup algorithms, suffix stripping algorithms, lemmatisation algorithms, $n$-gram analysis, and hybrid approaches. A stemming algorithm is assessed with the following two characteristics. Sometimes the algorithm brings several morphologically related words used in different context with different meaning to a single stem. Such false-positive predictions of the stemming algorithm are called overstemming and may worsen the discriminating abilities of the obtained feature set. In other cases, the algorithm does not bring various related words to one stem. Such false-negative predictions of the stemming algorithm are called understemming and may reduce its effect of dimensionality reduction. A well-optimised stemming algorithm...
provides an effective trade-off between overstemming and understemming. An example of stemming is given in Formula 2.1 – 2.3 and 2.16 – 2.20.

- **Word clustering based on dictionaries of synonyms.** The intuition behind this method is based on the assumption that all the synonyms of a word can be used interchangeably without causing any distortions in the general meaning of the text. The method aims to combine words into synonymic groups (clusters). The set of clusters is used as a new set of words (vocabulary), i.e., all the words belonging to one cluster are replaced by a single synonym. Word clustering was proposed by Brown et al. [1992] and applied to various fields, e.g., speech recognition, machine translation, named entity recognition, query expansion, text classification, and word-sense disambiguation. Momtazi and Klakow [2009] proposed using word clustering jointly with a cluster-based LM that succeeded in overcoming the problems of data sparsity and exact word matching in a small amount of text.

- **Part-of-speech tagging.** This filter replaces each word by its Part-of-Speech (POS) tag. POS tagging is useful for speaking style modelling and syntax modelling in particular [Batliner, Hacker, and Nöth, 2008]. POS tagging can be performed using rule-based and also stochastic LMs [Schmid, 1994]. In each case, the model predicts the POS tag of each word given its neighbouring context. For example, one of the rules of a rule-based LM for English POS tagging could be as follows: "a word following "the" in English is most likely a noun". An output example of a POS tagger is shown in Figure 2.2.

- **Dependency parsing.** This filter replaces each word by its dependency tag. Similarly to POS tagging, dependency parsing is useful for speaking style modelling and particularly for syntax modelling. Likewise POS tagging, dependency parsing can be performed using rule-based as well as stochastic LMs. An output example of a dependency parser is depicted in Figure 2.2. A POS tag is given under each word: DET – definite article, NOUN – noun, VERB – verb, ADV – adverb, ADJ – adjective. A dependency tag is specified under each arrow and belongs to the word the arrow points at: the main verb ("is"), which has no incoming arrows, is denoted as ROOT (this tag was omitted in the figure); a determiner (det) is the relation between the head of a noun phrase ("parcel") and its determiner ("the"); a nominal subject (nsubj) is a noun phrase ("parcel") which is the syntactic subject of a clause; an adjectival complement (acomp) of a verb ("is") is an adjectival phrase ("heavy") which functions as the complement (like an object of the verb); an adverb modifier (advmod) of a word ("heavy") is a (non-clausal) adverb or adverb-headed phrase ("too") that serves to modify the meaning of the word. Detailed tag notations and algorithm descriptions can be found in [De Marneffe and Manning, 2008; Spacy 2019].

**Term extraction (vocabulary forming).** At this preprocessing stage, unique textual units are extracted from text. One usually understands textual units (terms) as separate words (unigrams), though they are nothing but a special case of n-grams, and therefore other cases can also be considered. All the unique textual units form a vocabulary. After that, each term occurrence in the text is replaced by the index of the term in the vocabulary. Unknown terms, which are absent in the training data, need to be removed from test data at this stage since no numeric representations have been learnt for them.

**Term weighting.** At this preprocessing stage, each term receives a weight (real value) reflecting the importance of the term in the context of a given classification task.
All the term weighting methods considered in the present study are based on the Bag-of-Words (BOW) model [Y. Zhang et al., 2010]. An early reference to BOW in a linguistic context can be found in [Harris, 1954]. This is a relatively simple approach to text representation in which a document is represented as a vector of term occurrence counts. The length of this vector is equal to the vocabulary size. In other words, if a term appears in a particular document, the corresponding component of the document vector is set to be equal to the number of occurrences of the term in the document. Otherwise, the component is set to be equal to zero. An example of such a model is given below. Suppose that our training data set consists of the following documents:

\[\text{doc}_1 = "Mike likes to play football."},
\text{doc}_2 = "Jonny likes playing football too.",
\text{doc}_3 = "Mary doesn't like playing football. She likes playing the guitar.".

(2.1) (2.2) (2.3)

No linguistic filtering is used. After tokenisation and unigram extraction, we obtain the following:

\[\text{doc}_1 = "mike likes to play football"},
\text{doc}_2 = "jonny likes playing football too",
\text{doc}_3 = "mary doesn't like playing football she likes playing the guitar".

(2.4) (2.5) (2.6)

\[V = \{"mike","likes","to","play","football","jonny","playing","too","mary","doesnt","like","she","the","guitar"\},\]

(2.7)

\[B = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}.\]

(2.8)

\(V\) is a set of unique terms (vocabulary), \(B_{i,j}\) is the number of occurrences of the \(j\)-th term in the \(i\)-th document. A similar model built for bigrams looks as follows:
2 Background and Related Work

\[ V = \{ \text{"mike likes"}, \text{"likes to"}, \text{"to play"}, \text{"play football"}, \text{"jonny likes"}, \text{"likes playing"}, \text{"playing football"}, \text{"football too"}, \text{"mary doesn't"}, \text{"doesn't like"}, \text{"like playing"}, \text{"football she"}, \text{"she likes"}, \text{"playing the"}, \text{"the guitar"} \} \]  

\[ B = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \]  

The dimensionality (vocabulary size) of the bigram model turns out to be higher than the dimensionality of the unigram one. Generally, the greater the \( n \) value is, the longer context the model covers, the higher vector space dimensionality the model has, and the more data we need to train this model for sufficient generalisation.

The vector space obtained after applying a linguistic filter tends to have a lower dimensionality compared to the space specified above. After tokenisation, stop-word filtering, and unigram extraction, we obtain the following:

\[ doc_1 = \text{"mike likes play football"}, \]  
\[ doc_2 = \text{"jonny likes playing football"}, \]  
\[ doc_3 = \text{"mary like playing football likes playing guitar"}. \]

\[ V = \{ \text{"mike"}, \text{"likes"}, \text{"play"}, \text{"football"}, \text{"jonny"}, \text{"playing"}, \text{"mary"}, \text{"like"}, \text{"guitar"} \} \]  

\[ B = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \ 0 & 1 & 0 & 1 & 0 & 2 & 1 & 1 \end{bmatrix}. \]  

We note that the stop-word filtering reduces the model dimensionality indeed. Instead of the stop-word filter, we can also apply stemming. After tokenisation, stemming, and unigram extraction, the resulting model looks as follows:
2.2 Feature Extraction

\[ \text{doc}_1 = "mike like play football", \]
\[ \text{doc}_2 = "jonny like play football too", \]
\[ \text{doc}_3 = "mary do like play football she like play the guitar". \]

\[ V = \{ "mike", "like", "play", "football", "jonny", "too", "mary", "do", "she", "the", "guitar" \}, \]
\[ B = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 2 & 2 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{bmatrix}. \]

The joint use of stop-word filtering and stemming can further improve the effect of dimensionality reduction. After tokenisation, stop-word filtering, stemming, and unigram extraction, we obtain the following:

\[ \text{doc}_1 = "mike like play football", \]
\[ \text{doc}_2 = "jonny like play football"., \]
\[ \text{doc}_3 = "mary play football like play guitar". \]

\[ V = \{ "mike", "like", "play", "football", "jonny", "mary", "guitar" \}, \]
\[ B = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 1 & 1 & 1 & 0 & 0 \\
3 & 0 & 2 & 1 & 0 & 1 & 1 \\
\end{bmatrix}. \]

Although the semantic meaning of the text has been distorted after applying the linguistic filters, it is still an open question if the linguistic filtering also affects the speaking style that we try to classify for AD.

We can directly use any of the aforementioned BOW models for feature extraction and classification, although they all have a critical drawback: all the terms in the vocabulary are assumed to be of equal importance for classification. This is obviously not the case in real-life applications, e.g., a system performing topic detection in the domain of technical support should assign higher weights for the words "notebook", "internet", "smartphone", "display", etc., than for the words "problem", "issue", "service", "cat" etc., since we may face any of them in the context of various topics and the latter word does not even belong to the domain of technical support. Therefore, term weighting is required; this procedure emphasises more important terms for classification by assigning higher weights to them.

There exist two categories of term weighting methods: unsupervised and supervised. \textit{Unsupervised term weighting methods} do not use any information about the class labels of the documents introduced in a training set. The weights are calculated given only the statistics of term occurrences in the training set. On the one hand, such an unsupervised term weighting method can be trained on a large amount of out-of-domain unlabelled data which is quite easy to obtain. On the other hand, the resulting model is likely to be not that finely tuned to the particular text classification task compared...
to a *supervised term weighting method* which considers target label occurrence statistics alongside with the statistics of term occurrences in the training set.

As a rule, term weighting is a product of two parts. The first one represents the statistics of term occurrences in a given document, while the second part estimates the statistics of term occurrences in the entire training set. In our study, the first and the second part are calculated as Term Frequency (TF) and various modifications of Inverse Document Frequency (IDF) respectively.

The TF part is computed as follows:

\[
TF_{ij} = \log(1 + tf_{ij}),
\]

\[
tf_{ij} = \frac{n_{ij}}{N_j},
\]

where \(n_{ij}\) is the number of times the \(i\)-th term occurs in the \(j\)-th document and \(N_j\) is the length (number of terms) of the \(j\)-th document. Here, the term frequency is adjusted to the document length and logarithmically scaled since the number of repetitive terms in a document tends to be significantly lower than the length of the document. We combine this TF part with each of the IDF parts described below.

The second part of term weighting is computed once for each term in the vocabulary and hence does not depend on a specific document being classified. We calculate this part in the following seven different ways: Inverse Document Frequency (IDF), Gain Ratio (GR), Confident Weights (CW), Term Second Moment (TM2), Relevance Frequency (RF), Term Relevance Ratio (TRR), and Novel Term Weighting (NTW). In fact, all these methods are different modifications of IDF.

- **Inverse Document Frequency.** This very popular unsupervised approach to term weighting was proposed by Salton and Buckley [1988]. The method has the following versions:
  1. **IDF_1:**

\[
IDF_i = \log \frac{|D|}{n_i},
\]

where \(D\) is a training set of documents and \(n_i\) is the number of documents containing the \(i\)-th term.

2. **IDF_2:** This version also uses Formula 2.28. However, \(n_i\) is computed as the number of times the \(i\)-th term appears in the training set.

3. **IDF_3:**

\[
IDF_i = \left( \frac{|D|}{n_i} \right)^\alpha,
\]

\[
\alpha \in (0, 1),
\]

where \(n_i\) is computed similarly to IDF_1 and \(\alpha\) is a parameter.

In our study, we use the most popular version IDF_1.

- **Gain Ratio.** This supervised method is usually applied to term selection [Yang and Pedersen, 1997], though Debole and Sebastiani [2004] has shown that the method can also be used for term weighting since it estimates term importance in a certain sense. GR weights are computed as follows:
where \( P(t,c) \) is the estimation of the probability that a document contains the term \( t \) and belongs to the category \( c \), \( P(t) \) is the estimation of the probability that a document contains the term \( t \), \( P(c) \) is the estimation of the probability that a document belongs to the category \( c \). Then, the weight of the term \( t_i \) is computed as the maximum over all target classes as follows:

\[
GR(t_i) = \max_{c_j \in C} GR(t_i, c_j),
\]

where \( C \) is the set of all classes.

- **Confident Weights.** This supervised approach to term weighting was proposed by Soucy and Mineau [2005]. According to the CW method, the term \( t \) has a non-zero weight for the class \( c \) only if the frequency of the term \( t \) in documents belonging to the class \( c \) is greater than the frequency of the same term in all other classes. First, the proportion of documents containing the term \( t \) is defined by the Wilson estimate [Wilson, 1927] \( p(x, n) \) as follows:

\[
p(x, n) = \frac{x + 0.5z_{\alpha/2}^2}{n + z_{\alpha/2}^2},
\]

where \( x \) is the number of documents containing the term \( t \) in a training set, \( n \) is the total number of documents in this training set, and \( z_{\alpha/2} \) satisfies the equation \( \Phi(z_{\alpha/2}) = \alpha/2 \). Here, \( \Phi \) is the \( t \)-distribution function, also known as Student’s law, when \( n < 30 \) and the normal distribution function when \( n \geq 30 \). In the present work, \( \alpha \) and \( 0.5z_{\alpha/2}^2 \) equal 0.95 and 1.96 respectively (as recommended by Soucy and Mineau [2005]). Let us denote a confidence interval for \( p \) as \( (\underline{p}, \overline{p}) \). The limits of this interval are defined as follows:

\[
\underline{p} = p - 0.5z_{\alpha/2}^2 \sqrt{\frac{p(1-p)}{n + z_{\alpha/2}^2}},
\]

\[
\overline{p} = p + 0.5z_{\alpha/2}^2 \sqrt{\frac{p(1-p)}{n + z_{\alpha/2}^2}}.
\]

Most categorisation tasks can be resolved using binary classifiers only, i.e., each classifier determines whether a document belongs to a specific category or not. Therefore, a task with \( C \) categories can be resolved with \( C \) binary classifiers. For a given category \( c \), let us denote Formula 2.33 applied to the positive documents (which belong to this category) as \( p_+ \). Otherwise (for the documents which do not belong to this category), let us denote this formula as \( p_- \). After that, the strength of the term \( t \) in the category \( c \) is defined in the following way:

\[
str(t, c) = \begin{cases} 
\log_2 \left( \frac{2p_+}{p_+ + p_-} \right) & \text{if } p_+ > p_-,
0 & \text{otherwise}.
\end{cases}
\]

\( \text{str}(t, c) \)
The CW weight of the term \( t \) is equal to its maximum strength which is computed as follows:

\[
CW(t) = \left( \max_{c \in C} str(t, c) \right)^2.
\]  

(2.37)

The numerical experiments conducted by Soucy and Mineau [2005] have shown a classification performance improvement that the CW method combined with Support Vector Machine (SVM) and \( k \) Nearest Neighbours (k-NN) demonstrates over the IDF and GR term weighting schemes on some benchmark data sets.

- **Term Second Moment.** This supervised term weighting method was proposed by Xu and Li [2007]. Let \( P(c_j | t_i) \) be an empiric estimation of the probability that a document belongs to the category \( c_j \) with the condition that the document contains the term \( t_i \) and let \( P(c_j) \) be an empiric estimation of the probability that the document belongs to the category \( c_j \) without any conditions. The idea behind computing these probability estimations is that their difference reflects the importance of the term. Therefore, the weight of the term \( t_i \) can be calculated as follows:

\[
TM(t_i) = \left| \sum_{j=1}^{|C|} \left( P(c_j | t_i) - P(c_j) \right)^2 \right|,
\]  

(2.38)

where \( C \) is the set of all classes.

- **Relevance Frequency.** This supervised approach to term weighting was proposed by Lan et al. [2008]. The idea behind the RF method is that the greater is the difference between the concentration of a term with a high frequency in the positive category and its concentration in the negative category, the more the term contributes to the detection of this category. RF term weights are computed as follows:

\[
RF(t_i, c_j) = \log_2 \left( 2 + \frac{a_{ij}}{\max \{1, a_{ij}\}} \right),
\]  

(2.39)

\[
RF(t_i) = \max_{c_j \in C} RF(t_i, c_j),
\]  

(2.40)

where \( a_{ij} \) is the number of documents belonging to the category \( c_j \) which contain the term \( t_i \) and \( \pi_j \) is the number of documents, which also contain this term, in all other categories.

The experimental results obtained by Lan et al. [2008] demonstrate that the RF method in combination with the SVM and k-NN classification algorithms surpasses the IDF, GR, CW, and TM2 term weighting schemes in terms of classification performance.

- **Term Relevance Ratio.** The TRR term weights proposed by Ko [2012] are computed as follows:
\[ TRR(t_i, c_j) = \log_2 \left( 2 + \frac{P(t_i|c_j)}{P(t_i|\overline{c_j})} \right), \quad (2.41) \]

\[ P(t_i|c_j) = \frac{|\mathcal{T}_{c_j}|}{|V|} \frac{\sum_{k=1}^{L} t_{f_{ik}}}{\sum_{l=1}^{L} \sum_{k=1}^{L} t_{f_{lk}}}, \quad (2.42) \]

\[ P(t_i|\overline{c_j}) = \frac{|\mathcal{T}_{\overline{c_j}}|}{|V|} \frac{\sum_{k=1}^{L} t_{f_{ik}}}{\sum_{l=1}^{L} \sum_{k=1}^{L} t_{f_{lk}}}, \quad (2.43) \]

\[ TRR(t_i) = \max_{c_j \in C} TRR(t_i, c_j), \quad (2.44) \]

where \( t_i \) is a term, \( c_j \) is the class of a document, \( \overline{c_j} \) are all other classes, \( C \) is the set of all classes, \( V \) is the vocabulary of a training data set, \( \mathcal{T}_{c_j} \) is the document set of the class \( c_j \), \( \mathcal{T}_{\overline{c_j}} \) is the document set of all the other classes, and \( t_{f_{ij}} \) is specified in Formula 2.27.

According to the experiments conducted by Ko [2012], the TRR method in combination with the SVM and k-NN classifiers performs equally or better than the IDF, GR, CW, TM2, and RF term weighting schemes on several benchmark data sets.

- **Novel Term Weighting.** This method of supervised term weighting was proposed in [Gasanova et al., 2013; Gasanova et al., 2014]. NTW term weights are computed applying a modified formula of the fuzzy rule relevance estimation which is used for fuzzy classifiers [Ishibuchi et al., 1999]. The membership function has been replaced by term frequency in a given class. The computational procedure is as follows. Let \( L \) be the number of classes, \( n_i \) is the number of documents belonging to the \( i \)-th class, \( N_{ij} \) is the number of times the \( j \)-th term appears in all the documents of the \( i \)-th class. NTW weights are computed as follows:

\[ T_{ij} = \frac{N_{ij}}{n_i}, \quad (2.45) \]

\[ R_j = \max_i T_{ij}, \quad (2.46) \]

\[ S_j = \arg(\max_i T_{ij}), \quad (2.47) \]

\[ NTW_j = \frac{1}{L} \left( R_j - \frac{1}{L-1} \sum_{i=1}^{L} T_{ij} \right), \quad (2.48) \]

where \( T_{ij} \) is the relative frequency of occurrences of the \( j \)-th term in the \( i \)-th class, and \( S_j \) is the class which we assign to the \( j \)-th term.

The NTW weight of a term is higher if the term tends to appear in one class than if the term frequently appears in several classes. The maximum value of a term weight is equal to 1; it means that the term appears only in the documents of a single class. The minimum value of a term weight is equal to 0; it means that the term appears in all classes with similar frequencies.
**Dimensionality reduction (optional).** Data sets containing several thousands of unique terms are quite common in real-life applications. After term extraction and term weighting, the dimensionality of the obtained document vectors is equal to the dictionary size of a training data set. Some terms (usually those with the lowest weights) are highly likely to be irrelevant to a given classification task. At the same time, such terms (features) produce additional noise that confuses our classifier, and therefore there is the necessity of dimensionality reduction. There exist many approaches to dimensionality reduction. However, in this section we discuss only those which are typical for text processing. In general, approaches to dimensionality reduction can be divided into linguistic and numerical dimensionality reduction methods. These two categories are briefly described below.

- **Linguistic dimensionality reduction methods.** Methods belonging to this category remove or transform some linguistic information contained in text. Stemming and stop-word filtering are typical examples of such methods. They are usually language-dependent and applied prior to term extraction.

- **Numerical dimensionality reduction methods.** Such methods are usually applied directly after term weighting. They perform dimensionality reduction in a general sense and do not depend on the nature of the features being considered. This category of methods includes Principal Component Analysis (PCA), Recursive Feature Elimination (RFE), and intramodel feature selection and feature transformation. These approaches are described in Section 2.3.

- **Feature selection based on term weights.** The most straightforward approach to numerical dimensionality reduction is to use IDF coefficients as feature weights since these coefficients reflect the importance of each term regarding a given classification task. Therefore, we can remove a certain percentage of terms with the lowest weights by ignoring these terms during the classification process. However, this percentage should be fine-tuned since any feature selection method always causes information losses; we want these losses to include only noise and no valuable information that may be relevant to the classification task.

- **Feature Transformation Based on Terms Belonging to Classes.** This feature transformation method was proposed by Sergienko [2016]. We denote it as FT for simplicity. Its underlying idea is that each term in the vocabulary can be assigned to the most appropriate category of a classification task. As a result, the method allows us to reduce the dimensionality of the text classification problem radically to the number of target classes. Such an assignment is automatically conducted within some term weighting methods. The most appropriate class $\kappa_i$ from the set of all target classes $C$ can be set for the term $t_i$ using GR weights as follows:

\[
\kappa_i = \arg \max_{c_j \in C} GR(t_i, c_j),
\]

(2.49)

where $GR(t_i, c_j)$ is computed using Formula 2.31.

A similar approach can be applied to CW weights:

\[
\kappa_i = \arg \max_{c_j \in C} str(t_i, c_j),
\]

(2.50)

where $str(t_i, c_j)$ is computed by Formula 2.36.

It can also be used in combination with RF weights:
\[ \kappa_i = \arg\max_{c_j \in C} RF(t_i, c_j), \]  
\hspace{1cm} (2.51) 

where \( RF(t_i, c_j) \) is computed using Formula 2.39.

It can be applied to TRR weights in a similar way:

\[ \kappa_i = \arg\max_{c_j \in C} TRR(t_i, c_j), \]  
\hspace{1cm} (2.52) 

where \( TRR(t_i, c_j) \) is computed by Formula 2.41.

The procedure for NTW weights is slightly different:

\[ \kappa_i = \arg\max_{c \in C} \frac{n_{ic}}{N_c}, \]  
\hspace{1cm} (2.53) 

where \( n_{ic} \) is the number of documents belonging to the \( c \)-th class which contain the \( i \)-th term, and \( N_c \) is the number of all documents in the \( c \)-th class. In other words, we choose the class with the maximum relative frequency of the term among all the target classes. Formula 2.53 can also be applied to the remaining term weighting methods (IDF and TM2).

As soon as each term has been assigned to its most appropriate class, we apply the Algorithm 1 of feature transformation based on terms belonging to classes.

**input**: vector of term weights \( w \), vector of the most appropriate term classes \( c \), number of all target classes \( C \)

**output**: feature vector \( s \) after feature transformation

\begin{algorithm}
  \begin{algorithmic}
    \For {\( i = 1 \) to \( C \)}
    \State \( s_i \leftarrow 0; \)
    \EndFor
    \For {\( i = 1 \) to \( \text{length}(w) \)}
    \State \( s_{c_i} \leftarrow s_{c_i} + w_i; \)
    \EndFor
    \Return \( s \)
  \end{algorithmic}
\end{algorithm}

**Algorithm 1**: Feature transformation based on terms belonging to classes.

The FT method described above significantly reduces the dimensionality of a text classification problem without time-consuming calculations which are required for the PCA algorithm. Therefore, the FT method is a very efficient tool for real-time utterance classification, e.g., in SDSs.

**Distributed Word Representations**

Distributed word representations were used for text-based AD by many researchers, e.g., S. V. Ravuri and Stolcke [2014], S. Ravuri and Stolcke [2015], Ouchi and Tsuboi [2016], and R. Zhang et al. [2018]. Any unsupervised method for learning distributed word representations relies on the statistics of word co-occurrences in a corpus. These statistics tend to be the primary source of information available to the method. Such a method is supposed to solve the following two problems: how meaning is generated from these statistics, and how the resulting word vectors might represent
that meaning. Two of the most popular methods for learning distributed word representations are the skip-gram model [Mikolov et al., 2013] and GloVe [Pennington et al., 2014]. The name of the latter method stands for Global Vectors since global corpus statistics are captured directly by the model. We make our choice in favour of the GloVe method as it is purely based on matrix factorisation and therefore has lower computational complexity compared to the process of training a skip-gram neural network model while giving approximately the same or even better result [Pennington et al., 2014]. The GloVe method is described in detail in this section below.

First, we define some notation. Let $X$ be a matrix of word-word co-occurrence counts. Its element $X_{ij}$ indicates the number of times the $j$-th word appears in the context of the $i$-th word. Let $X_i = \sum_k X_{ik}$ be the number of times any word occurs in the context of the $i$-th word. Finally, let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that the $j$-th word appears in the context of the $i$-th word. It is also necessary to specify the length of the context window and whether to distinguish forward context from backward context. Pennington et al. [2014] use a decreasing weighting function, so that word pairs that are $d$ words apart contribute $1/d$ to the total count; the word pair does not contribute to the total count at all if $d > 10$. Both forward and backward context are considered.

In a simple example, we are going to show how certain aspects of meaning can be extracted directly from co-occurrence probabilities. Consider two words $i$ and $j$ that exhibit a particular aspect of interest; for concreteness, suppose we are interested in the concept of thermodynamic phase, for which we might take $i = \text{ice}$ and $j = \text{steam}$. The relationship between these two words can be determined by studying the ratio of their co-occurrence probabilities with various probe words $k$.

For words $k$ related to $i$ but not $j$, e.g., $k = \text{solid}$, the ratio $P_{ik}/P_{jk}$ is expected to be large. Otherwise, for words $k$ related to $j$ but not $i$, e.g., $k = \text{gas}$, the ratio is supposed to be small. For words $k$ like $\text{water}$ or $\text{fashion}$, which are either related to both $i$ and $j$, or to neither, the ratio is expected to be close to one. Table 2.1 shows these probabilities and their ratios for a large corpus, and the values confirm our expectations. Compared to the raw probabilities, the ratio is better able to distinguish relevant words ($\text{solid}$ and $\text{gas}$) from irrelevant words ($\text{water}$ and $\text{fashion}$) and it is also better able to discriminate between the two relevant words.

The statistics given above motivates us to choose ratios of co-occurrence probabilities rather than the probabilities themselves as the reference point for word vector learning. Given that the ratio $P_{ik}/P_{jk}$ depends on three words $i$, $j$, and $k$, the most abstract model can be described as follows:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}},$$

(2.54)

Table 2.1: Co-occurrence probabilities of the target words $\text{ice}$ and $\text{steam}$ with some selected context words. The statistics are extracted from a large textual corpus containing approximately six billion tokens.

<table>
<thead>
<tr>
<th>Probability and ratio</th>
<th>$k = \text{solid}$</th>
<th>$k = \text{gas}$</th>
<th>$k = \text{water}$</th>
<th>$k = \text{fashion}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(k</td>
<td>\text{ice})$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$6.6 \times 10^{-5}$</td>
<td>$3.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>\text{steam})$</td>
<td>$2.2 \times 10^{-5}$</td>
<td>$7.8 \times 10^{-4}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>\text{ice})/P(k</td>
<td>\text{steam})$</td>
<td>$8.9$</td>
<td>$8.5 \times 10^{-2}$</td>
</tr>
</tbody>
</table>
where \( w \in \mathbb{R}^d \) are word vectors and \( \tilde{w} \in \mathbb{R}^d \) are separate context word vectors. The right side of Formula 2.54 is extracted directly from the corpus. The function \( F \) on the left side may depend on some as-of-yet unspecified parameters. We are going to make a unique choice regarding the function \( F \) that will satisfy some specific requirements, although this function can take many different forms.

We would like \( F \) to encode the information represented by the ratio \( P_{ik}/P_{jk} \) in the word vector space. The first constraint that we apply to \( F \) is connected with vector differences. We want the vector difference operation in our word vector space to make sense, and therefore we confine to those functions \( F \) depending only on the difference between the two target words. Taking this constraint into account, we can modify Formula 2.54 in the following way:

\[
F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}. \tag{2.55}
\]

The arguments of the function \( F \) in Formula 2.55 are vectors, while the right side is a scalar. \( F \) could be chosen to be a complicated function parametrised by, e.g., an Artificial Neural Network (ANN). However, this choice would probably obfuscate the linear structure of the word vector space that we are trying to capture. In order to avoid this problem, we apply the second constraint to Formula 2.55: we take a dot product of the arguments of \( F \) as follows:

\[
F \left( (w_i - w_j)^T \tilde{w}_k \right) = \frac{P_{ik}}{P_{jk}}. \tag{2.56}
\]

This decision allows us to prevent the function \( F \) from mixing the vector dimensions in undesirable ways. Next, we note that the roles of a word and a context word are interchangeable in word-word co-occurrence matrices. Therefore, our model should be irrelevant to the exchanges \( w \leftrightarrow \tilde{w} \) and \( X \leftrightarrow X^T \). However, Formula 2.56 is sensitive to them. The symmetry can be reached in two steps. First, we require the function \( F \) to be a homomorphism between the groups \((\mathbb{R}, +)\) and \((\mathbb{R}_{>0}, \times)\):

\[
F \left( (w_i - w_j)^T \tilde{w}_k \right) = F \left( w_i^T \tilde{w}_k \right) \cdot F \left( w_j^T \tilde{w}_k \right). \tag{2.57}
\]

Formula 2.57 in combination with Formula 2.56 gives the following:

\[
F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{2.58}
\]

The solution to Formula 2.57 is \( F = \exp \). Therefore, after finding the logarithm of each part of Formula 2.58 we obtain the following:

\[
w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i). \tag{2.59}
\]

Next, we note that all the parts of Formula 2.59 except the term \( \log(X_i) \) demonstrate the exchange symmetry. However, \( \log(X_i) \) does not depend on \( k \), and therefore it can be absorbed into a bias \( b_i \) for \( w_i \). Finally, adding an additional bias \( \tilde{b}_k \) for \( \tilde{w}_k \) restores the symmetry as follows:

\[
w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}). \tag{2.60}
\]
Formula 2.60 is an essentially simplified version of Formula 2.54. However, the logarithm in Formula 2.60 diverges whenever its argument becomes zero. This issue can be resolved applying an additive shift to the logarithm argument: \( \log(X_{ik}) \rightarrow \log(1 + X_{ik}) \). This solution also maintains the sparsity of \( X \). However, Pennington et al. [2014] suggest another solution to this issue; the problem of the diverging logarithm is resolved multiplying the logarithm by a weighting function. Details of this solution are explained below. The main drawback of the obtained model is that it assigns equal weights to all co-occurrences. This issue becomes critical particularly for those co-occurrences which happen rarely or never. They bring a lot of noise to our textual data while carrying little useful information. Meanwhile, such zero values make 75–95% of all the elements in \( X \), depending on the vocabulary size and corpus.

Pennington et al. [2014] propose a new weighted least squares regression model that addresses the problems mentioned above. We can rewrite Formula 2.60 as follows:

\[
\sum_{i,j=1}^{V} f(X_{ij}) (w_{i}^T \hat{w}_{j} + b_{i} + \hat{b}_{j} - \log(X_{ij})) = 0. \tag{2.61}
\]

After casting Formula 2.61 to a least squares problem and introducing a weighting function \( f(X_{ij}) \) into the objective function, we obtain the following:

\[
J = \sum_{i,j=1}^{V} f(X_{ij}) (w_{i}^T \hat{w}_{j} + b_{i} + \hat{b}_{j} - \log(X_{ij})) ^2, \tag{2.62}
\]

where \( V \) is the vocabulary size. The weighting function \( f(X_{ij}) \) should possess the following properties:

1. The function \( f(x) \) should satisfy the constraint \( f(0) = 0 \). If \( f(x) \) is considered to be a continuous function, it should vanish as \( x \rightarrow 0 \) fast enough that \( \lim_{x \rightarrow 0} (f(x) \log^2 x) \) is finite. In this case, the divergence of \( \log(X_{ij}) \) when \( X_{ij} = 0 \) is no longer an issue.
2. The function \( f(x) \) should be non-decreasing so that rare co-occurrences are not overweighted.
3. The function \( f(x) \) should be relatively small for large values of \( x \) so that rare co-occurrences are not disregarded.

Although there exist many functions satisfying these constraints, Pennington et al. [2014] suggest to use the following class of functions that turned out to perform well:

\[
f(x) = \begin{cases} 
(x/x_{\text{max}})^{\alpha} & \text{if } x < x_{\text{max}}, \\
1 & \text{otherwise}. 
\end{cases} \tag{2.63}
\]

According to the experiments conducted by Pennington et al. [2014], the performance of the model insignificantly depends on the cutoff that is set to \( x_{\text{max}} = 100 \), and \( \alpha = 3/4 \) gives a small improvement over a linear increment with \( \alpha = 1 \).

By minimising the objective function \( J \) defined by Formula 2.62 with the AdaGrad algorithm [Duchi et al., 2011], Pennington et al. [2014] find two sets of word vectors: \( W \) and \( \tilde{W} \). These sets are equivalent and differ only due to their random initialisations since the matrix \( X \) is symmetric. The sums \( W + \tilde{W} \) are used as the final word vectors for better generalisation.
Fig. 2.3: Vector differences revealing a relationship between an adjective and its comparative and superlative forms.

An interesting output example of GloVe is given in Figure 2.3. In this example, the GloVe vector differences reveal a relationship between an adjective and its comparative and superlative forms. In the same example, the adjectives also form clusters by meaning. In a similar way, GloVe can capture relationships between other words, e.g., a relationship between the two genders represented with different words (man and woman, king and queen), a relationship between a company and its chief executive officer, a relationship between a city and its postal code, etc.

Let us summarise the key features of GloVe. The method possesses the following advantages:

- enables fast training;
- can be scaled to huge corpora;
- performs effectively even under a lack of training data and with low-dimensional word vectors;
- allows us to control the convergence precisely and to stop the training process anytime when improvements become small enough.

However, the approach also has a few disadvantages:

- uses a lot of memory since the collected co-occurrence statistics are global;
- may be sensitive to the initial learning rate.

### 2.2.4 ASR Metafeatures

The features belonging to this group describe various properties of the hypotheses generated by an ASR engine [Tsai et al., 2015]. Examples of these metafeatures are as follows:

- utterance duration;
- confidence of the best hypothesis;
• number of hypotheses;
• number of words in the best hypothesis.

The idea behind these attributes is based on the fact that machine-directed speech tends to match
the ASR acoustic and language patterns better compared to human-directed speech [Shriberg et al.,
2012].

2.2.5 Visual Features

When interacting with multiple speech-enabled devices, people tend to specify the desirable recipi-
ents of their spoken requests through eye gaze [Maglio et al., 2000; Brumitt and Cadiz, 2001]. The
same trend is observed in H-H-M conversations and underlies all visual AD methods [Skantze and
Gustafson, 2009]. The problem of video-based AD is usually reduced to determining whether a user
is looking at the system screen (On-View – Figure 2.4a) or not (Off-View – Figure 2.4b) while talk-
ing. The user’s focus of visual attention can be tracked by means of a frontal camera placed near the
screen of a mobile SDS [Batliner, Hacker, and Nöth, 2008].

Fig. 2.4: Two focuses of a user’s visual attention that were captured by a frontal camera placed above
the display of a mobile SDS.

However, there exists no direct relation between the On/Off-View and On/Off-Talk labels. On the
one hand, any SDS is perceived as a major situational attractor, and therefore people sometimes keep
looking at the system display even while talking among themselves. This phenomenon from social
psychology was studied by Argyle and Graham [1976]. On the other hand, people look very rarely
or do not look at the system at all in some other scenarios, e.g., while driving a car. In such situations,
humans stay visually focused on more important things than their in-car multimedia systems, i.e.,
on the driving process. The relation between the On/Off-View and On/Off-Talk labels turns out to
be extremely scenario-dependent. However, in the present study we confine only to those SDSs
talking to which users can easily switch their focus of visual attention. In this light, the only verbal
rule connecting the On/Off-View and On/Off-Talk labels together sounds as follows: *if a person is
talking and not looking at the SDS, he or she is highly likely to address another human. However, it
is impossible to make any reliable addressee prediction in case of On-View* [Batliner, Hacker, and
Nöth, 2008; Tsai et al., 2015). Similarly to the beamforming features, Tsai et al. [2015] show that using the visual modality in combination with others can be beneficial in some cases, although using this modality alone is not enough for reliable AD. Therefore, complex ML methods and particularly data fusion techniques are required to determine the exact relations between the On/Off-View and On/Off-Talk labels.

### Amount-of-Movement Features

Tsai et al. [2015] extract the three families of visual features listed below. The first family comprises features capturing the amount of movement. Their examples are as follows:

- variance of the speaker’s face location;
- variance of the speaker’s pose angle;
- average variance of all the participants’ face locations.

These attributes are computed throughout each utterance as well as on various time intervals of up to three seconds before and after the utterance. The intuition behind the amount-of-movement features is that humans stand more stationary while interacting with an SDS than when they address other people [Tsai et al., 2015].

### Face Orientation Features

The second family of visual features describes face orientation [Tsai et al., 2015]. Examples of the attributes belonging to this family are as follows:

- the speaker’s average pose angle in the up-down direction;
- the speaker’s average pose angle away from the computer in the left-right direction (normalised pose angle);
- fraction of the speaker’s pose-angle estimates that were unavailable when the person turned their face too far to the side.

These measurements are computed throughout each utterance as well as on various time intervals of up to three seconds before and after the utterance in order to account for lags between the moments when speech begins and when the face turns. The core idea of using this feature family is that humans tend to look at the addressees of their speech [Skantze and Gustafson, 2009]. According to this idea, eye tracking seems to be a better approach to monitoring a user’s focus of visual attention compared to head pose tracking. However, ordinary eye trackers using distant video cameras without any wearable devices, such as eye tracking glasses, are unable to provide reliable results at distances of 80 cm and longer [Tobii Technology AB 2019]. The short distance eye tracking is sufficient for seated desktop interfaces, though it is not for the standing interactions in the experiments conducted by Tsai et al. [2015].

### Physical Distance Features

The visual features belonging to the third family reflect physical distances between the participants [Tsai et al., 2015]. Examples of these attributes are as follows:
• distance between the speaker and the nearest/farthest participant;
• change in distance between the speaker and nearest/farthest participant over two neighbouring
time intervals.

For each feature, a single distance metric is measured over a time interval by calculating the mini-
mum, mean, and maximum of the corresponding frame-level distance values. As before, the mea-
surements are computed throughout each utterance as well as on various time intervals of up to three
seconds before and after the utterance. The intuition behind the third family of visual features is that
people standing far apart are less likely to have a joint discussion than those standing closer to each
other [Tsai et al., 2015].

2.2.6 System Features

System features contain information regarding the system state, including the dialogue manager
state [Tsai et al., 2015]. Examples of the attributes belonging to this group are as follows:
• number of participants involved in the interaction;
• time elapsed since the artificial conversational agent last spoke;
• dialogue act type of the artificial agent’s last utterance (question, confirmation, answer, etc.).

In contrast to most of the other features considered, some of the system attributes have a categorical,
rather than numerical, nature. The core idea of using these features is based on the fact that the
Off-Talk probability depends on the dialogue context in which a person speaks [Tsai et al., 2015].

2.3 Feature Selection and Feature Transformation Methods

The aim of any approach to dimensionality reduction is to reduce noise in the data and also to
combat the dimensionality curse in order to achieve more accurate predictions. Dimensionality re-
duction methods can be divided into feature selection and feature transformation approaches. The
idea behind feature selection is to remove highly correlated features and features which are irre-
elevant to the task being solved. The underlying idea of feature transformation is to obtain effective
lower-dimensional feature representations capturing as much data variability as possible.

The approaches to numerical dimensionality reduction described in the present section can be
applied to any features from Section 2.2 regardless of their nature.

2.3.1 Principal Component Analysis

Principal Component Analysis (PCA) [Jolliffe, 2011] is one of the most popular approaches to
numerical dimensionality reduction. This feature transformation method was proposed by Pearson
[1901]. The core idea of PCA is to apply an orthogonal linear transformation to convert a set of ob-
servations (which are described using possibly correlated variables) into a set of values represented
using linearly uncorrelated variables which are called principal components. This transformation
is defined in such a way that the first principal component has the largest possible variance (i.e.,
this component accounts for as much of the variability in the data as possible) and each succeeding
component in turn has the highest variance given the constraint that this component is orthogonal
Feature Selection and Feature Transformation Methods

2.3 Feature Selection and Feature Transformation Methods

...to all the preceding components. The resulting variables (each being a linear combination of the initial variables) form an uncorrelated orthogonal basis set. The idea behind PCA is illustrated in Figure 2.5. PCA is sensitive to the relative scaling of the initial variables.

Consider a data matrix $X$ consisting of $n$ rows (data examples) and $p$ columns (features). PCA provides effective results when $p \ll n$. We assume that each feature in $X$ has already been brought to zero mean. Let $W$ be a matrix of $l$ rows (that equals the dimensionality of the new feature space) and $p$ columns (that equals the dimensionality of the initial feature space) which transforms the example $x(i)$ to its new shape $t(i)$ in a new feature space. The values $p$ and $l$ always satisfy $l \leq p$, though $l$ is usually chosen given that $l \ll p$ in order to provide a noticeable effect of dimensionality reduction. This can be formally described as follows:

$$ x(i) = (x_1, \ldots, x_p)(i) \quad \text{for } i = 1, \ldots, n, $$

$$ w(k) = (w_1, \ldots, w_p)(k) \quad \text{for } k = 1, \ldots, l, l \leq p, $$

$$ t(i) = (t_1, \ldots, t_l)(i) \quad \text{for } i = 1, \ldots, n, $$

$$ t(i) = x(i)W^T \quad \text{for } i = 1, \ldots, n. $$

In order to maximise the variance of the vectors $t(i)$, $w(1)$ (the first vector of the new basis) can be found in the following way:

$$ w(1) = \arg \max_{\|w\|=1} \left( \|Xw^T\|^2 \right) = \arg \max_{\|w\|=1} \left( wX^T Xw^T \right). $$

Since $w(1)$ has been defined to be a unit vector, it also satisfies the following:
The quantity to be maximised can be recognised as a Rayleigh quotient. A standard result for a positive semi-definite matrix such as $X^T X$ is that the quotient’s maximum possible value is the largest eigenvalue of the matrix, which occurs when $w$ is the corresponding eigenvector.

The $k$-th component can be found by subtracting the first $k - 1$ principal components from $X$:

$$
\hat{X}_k = X - \sum_{s=1}^{k-1} Xw(s)w(s)^T,
$$

and then finding the vector $w(k)$ which extracts the maximum variance from the new data matrix $\hat{X}_k$:

$$
w(k) = \arg \max_{||w||=1} \left( \frac{||\hat{X}_k w||^2}{ww^T} \right) = \arg \max \left( \frac{w\hat{X}_k^T \hat{X}_k w^T}{ww^T} \right).
$$

It turns out that this gives the remaining eigenvectors of $X^T X$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues. Thus the weight vectors are eigenvectors of $X^T X$.

### 2.3.2 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is particularly useful for small data sets with a large number of features. This feature selection method requires a weighting coefficient to be assigned to each feature. Some parameters of a classification model, e.g., the components of the normal vector of a linear SVM or the synaptic weights of the inputs of an ANN, can be used as the feature weights for RFE. The model for obtaining the feature weights should be trained on the training set. After that, the features are sorted according to their weights prior to the first RFE iteration. A small group of features with the lowest weights is removed at each iteration. The classification performance of each reduced feature set in combination with some classification model is measured on a development set in terms of the target metric. A feature set is considered to be optimal if the classification performance starts dropping at subsequent iterations stably. The size of the optimal feature set should be fine-tuned on the development set.

### 2.3.3 Intramodel Dimensionality Reduction

Some mathematical models, e.g., ANNs, are capable of performing feature selection and feature transformation in an implicit manner inside the model. In particular, the synaptic weights of the inputs of a neural network are tuned during training in such a way that the network becomes more sensitive to certain features which turn out to be relevant to the task being solved. In other words, the network performs implicit feature selection inside the input layer.

One usually builds the structure of an ANN so that the feature space of a layer is transformed to a feature space of a smaller dimensionality in the subsequent layer. The information contained in these new features is more abstract compared to the features before the transformation. This can be interpreted as implicit feature transformation inside the model.
2.4 Classification Models

Most researchers approach AD as a classification problem. Classification models that we use in the present study are described in this section below.

2.4.1 Support Vector Machine

Support Vector Machines (SVMs, also known as Support Vector Networks) are supervised learning models with associated learning algorithms that can perform classification and regression [Cortes and Vapnik, 1995]. Despite their relative simplicity, such models are widely used in a lot of applications [L. Wang et al., 2005; Khan et al., 2010; Schuller et al., 2013; Schuller et al., 2015] and particularly appear as the baselines for some AD studies [Schuller et al., 2017; Siegert et al., 2019]. An SVM model divides data points in a vector space (examples) into two categories by a hyperplane so that the distance between the hyperplane and the nearest data point to it from either category is as large as possible. The data points are divided according to their belonging to the target class: in the ideal case, all data points of the class lie on the same side of the hyperplane, while data points belonging to any other classes lie on the other side. New examples are then mapped into that same space and predicted to belong to the target category based on the side of the hyperplane on which they fall.

The basic SVM can perform linear binary classification only. However, more advanced SVM-based models using a kernel-trick mechanism are capable of solving non-linear multiclass problems.

Suppose we have a training set \( S \) of \( n \) examples each of which consists of a \( p \)-dimensional feature vector \( x_i \) and a label scalar \( y_i \):

\[
S = \{(x_1, y_1), ..., (x_n, y_n)\},
\]

\[
x_i \in \mathbb{R}^p,
\]

\[
y_i \in \{1, -1\},
\]

where \( y_i \) indicates the class to which \( x_i \) belongs. Also suppose that the vectors \( x_i \) are normalised or standardised. Our aim is to find the "maximum-margin hyperplane" that divides the group of examples for which \( y_i = 1 \) from the group of examples for which \( y_i = -1 \). Furthermore, the hyperplane is defined in such a way that the distance between the hyperplane and the nearest data point from either group is maximised. Any hyperplane can be specified as the set of points \( x \) that satisfies the following equation:

\[
w^T x - b = 0,
\]

where \( w \) is the (not necessarily normalised) normal vector of the hyperplane.

**Hard-Margin Classification**

In case the training data is linearly separable, it is possible to find two parallel hyperplanes (Formula 2.76 and 2.77) that separate the two categories of the data so that the distance between the hyperplanes is maximised. The region bounded by these two hyperplanes in Figure 2.6 is called the
"margin", and the maximum-margin hyperplane (Formula 2.75) lies halfway between them. Given that $x$ is normalised or standardised, these two hyperplanes can be described as follows:

$$w^T x - b = 1,$$
$$w^T x - b = -1.$$  

Fig. 2.6: Maximum-margin hyperplane.

In order to maximise the distance between the two hyperplanes defined by Formula 2.76 and 2.77, which equals $2/\|w\|$, we need to minimise $\|w\|$. Equivalently, we can minimise $w^T w$ instead. We also need to introduce the following constraints in order to ensure that each data point lies on the correct side of the margin and prevent data points from falling into the margin:

$$w^T x_i - b \geq 1 \quad \text{for } i = 1, \ldots, n, \text{if } y_i = 1,$$
$$w^T x_i - b \leq -1 \quad \text{for } i = 1, \ldots, n, \text{if } y_i = -1.$$  

The constraints specified in Formula 2.78 and 2.79 can be rewritten as follows:

$$y_i (w^T x_i - b) \geq 1 \quad \text{for } i = 1, \ldots, n.$$  

This new constraint should be satisfied while minimising the value $w^T w$. The values $w$ and $b$ form the solution of the optimisation problem and also determine our hard-margin SVM classifier:

$$x \mapsto \text{sgn}(w^T x - b).$$
It is important to note that the maximum-margin hyperplane is completely defined by the data points \( x_i \) lying closest to it. These points are called support vectors.

**Soft-Margin Classification**

Let us now extend the SVM classifier for a more general case when the training data is linearly inseparable. For this purpose, we introduce the hinge loss function:

\[
\max(0, 1 - y(w^T x - b)),
\]

(2.82)

where \( y \) is the current target, \( x \) is the model input, and \( w^T x - b \) is the model output. This loss function becomes zero if the constraint given in Formula 2.80 is satisfied. Otherwise, the loss function value is proportional to the distance between the data point and the margin. We wish to minimise the following sum:

\[
\left[ \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i - b)) \right] + \lambda w^T w,
\]

(2.83)

where the parameter \( \lambda \) defines the trade-off between increasing the margin size and ensuring that the data points lie on the correct side of the margin. In general, \( \lambda \) can be interpreted as the tolerance to misclassification. This parameter should be fine-tuned on a development set. For small \( \lambda \) values, Formula 2.83 will behave similarly to the hard-margin SVM classifier if the input data is linearly separable.

**Training Process**

Let us now compute the parameters of the soft-margin SVM classifier. Minimising Formula 2.83 can be rewritten as a constrained optimisation problem with a differentiable objective function. After introducing a variable \( \zeta_i \), we can formulate the optimisation problem as follows:

\[
\zeta_i = \max(0, 1 - y_i(w^T x_i - b)),
\]

(2.84)

\[
\text{minimise } \frac{1}{n} \sum_{i=1}^{n} \zeta_i + \lambda w^T w,
\]

(2.85)

subject to \( y_i(w^T x_i - b) \geq 1 - \zeta_i \) and \( \zeta_i \geq 0 \), for \( i = 1, \ldots, n \).

(2.86)

This optimisation task is called the primal problem. Solving the Lagrangian dual of this problem, we obtain the simplified optimisation problem, which is also called the dual problem:

\[
\text{maximise } f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (x_i^T x_j) y_j c_j,
\]

(2.87)

subject to \( \sum_{i=1}^{n} c_i y_i = 0 \) and \( 0 \leq c_i \leq \frac{1}{2n\lambda} \) for \( i = 1, \ldots, n \).

(2.88)

The objective function (Formula 2.87) is a quadratic function of \( c_i \) and all the constraints (Formula 2.88) depend on \( c_i \) linearly. Therefore, the dual optimisation problem can be efficiently re-
solved using quadratic programming algorithms. It is important to note that the variables \( c_i \) are defined in such a way that:

\[
\mathbf{w} = \sum_{i=1}^{n} c_i y_i \mathbf{x}_i.
\]  

(2.89)

Furthermore, \( c_i = 0 \) exactly when \( \mathbf{x}_i \) lies on the correct side of the margin, and \( 0 < c_i < (2n\lambda)^{-1} \) when \( \mathbf{x}_i \) lies on the margin boundary. Therefore, \( \mathbf{w} \) can be represented by a linear combination of the support vectors. Given that \( y_i = \pm 1 \) and hence \( y_i = y_i^{-1} \), the parameter \( b \) can be recovered by finding a vector \( \mathbf{x}_i \) on the margin boundary and solving the following equation:

\[
1 - y_i (\mathbf{w}^T \mathbf{x}_i - b) = 0.
\]  

(2.90)

Finally, we obtain the value of the parameter \( b \):

\[
b = \mathbf{w}^T \mathbf{x}_i - y_i.
\]  

(2.91)

There also exist alternative approaches to computing the parameters of the soft-margin SVM classifier. Compared to the traditional method based on quadratic programming, they tend to perform faster on high-dimensional classification tasks with huge data sets. Given that the objective function specified in Formula 2.83 is a convex function of \( \mathbf{w} \) and \( b \), it is possible to apply sub-gradient descent methods to minimise it [Shalev-Shwartz et al., 2011]. Alternatively, coordinate descent methods can be applied to solve the dual problem defined in Formula 2.87 and 2.88. In this case, the coefficient \( c_i \) is iteratively adjusted in the direction of \( \partial f / \partial c_i \) for \( i = 1, \ldots, n \). After that, the resulting vector of coefficients \( (c^*_1, \ldots, c^*_n) \) is projected onto the nearest vector of coefficients that satisfies the given constraints. As a rule, Euclidean distances are used. This procedure is repeated until a near-optimal vector is found [Hsieh et al., 2008].

**Non-Linear Classification**

Let us now perform non-linear classification using the kernel-trick mechanism. A kernel trick enables the classification of data points which are linearly inseparable in the original vector space but will probably be linearly separable in a new transformed vector space of a higher dimensionality. The "trick" is that kernel methods represent the data only through a set of pairwise similarity comparisons between the original data observations \( \mathbf{x}_i \) (with the original coordinates in the lower-dimensional space), instead of explicitly applying a non-linear transformation \( \phi(\mathbf{x}_i) \) and representing the data by these transformed coordinates in the higher-dimensional space. Disregarding the kernel trick results in huge computational costs for high-dimensional tasks. First, we need to define the function \( \phi \) that will be used for linear classification in the space of the transformed data points \( \phi(\mathbf{x}_i) \). Furthermore, we need to specify a kernel function \( k \) that satisfies the following:

\[
k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j).
\]  

(2.92)

In other words, our kernel function acts as a modified dot product. The role of a kernel function in the classification process is illustrated in Figure 2.7. In this particular example, the following
transformation is used in order to make data points linearly separable in a two-dimensional space:

\[ \varphi(x) = \varphi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}, \]

(2.93)

where \( x_1 \) is the coordinate of the data points in Figure 2.7a. As a result, we obtain the following kernel function:

\[ \varphi(x) \cdot \varphi(x^*) = x_1 x_1^* + x_1^2 x_1^*^2 = (x \cdot x^*) + (x \cdot x^*)^2 = k(x,x^*), \]

(2.94)

where \( x \) and \( x^* \) are any two data points from Figure 2.7a. This function is just one of many possible kernel functions mapping our data into a linearly separable form in a higher-dimensional space. We continue our reasoning without connections to any specific kernel functions.

Similarly to Formula 2.89, the vector \( w \) in the transformed space satisfies the following:

\[ w = \sum_{i=1}^{n} c_i y_i \varphi(x_i), \]

(2.95)

where the coefficients \( c_i \) are obtained by solving the following dual optimisation problem, which is similar to that one specified in Formula 2.87 and 2.88:
maximise $f(c_1, ..., c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\varphi(x_i) \cdot \varphi(x_j)) y_j c_j = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i k(x_i, x_j) y_j c_j,$

subject to $\sum_{i=1}^{n} c_i y_i = 0$ and $0 \leq c_i \leq \frac{1}{2n\lambda}$, for $i = 1, ..., n$. (2.96)

The coefficients $c_i$ can be determined using quadratic programming as before. Again, we can find some index $i$, such that $0 < c_i < (2n\lambda)^{-1}$, so that $\varphi(x_i)$ lies on the boundary of the margin in the transformed space, and then find the parameter $b$:

$$b = w \cdot \varphi(x_i) - y_i = \left[ \sum_{j=1}^{n} c_j y_j \varphi(x_j) \cdot \varphi(x_i) \right] - y_i = \left[ \sum_{j=1}^{n} c_j y_j k(x_j, x_i) \right] - y_i.$$ (2.98)

Finally, we can determine our non-linear soft-margin SVM classifier:

$$z \mapsto \text{sgn}(w \cdot \varphi(z) - b) = \text{sgn} \left( \sum_{i=1}^{n} c_i y_i \varphi(x_i) \cdot \varphi(z) - b \right) = \text{sgn} \left( \sum_{i=1}^{n} c_i y_i k(x_i, z) - b \right).$$ (2.99)

There exists a great variety of kernel functions for specific problems. The most popular of them are as follows:

$$k_{\text{dot}}(x_i, x_j) = x_i \cdot x_j + c,$$ (2.100)

$$k_{\text{polynomial}}(x_i, x_j) = (\alpha x_i \cdot x_j + c)^d,$$ (2.101)

$$k_{\text{radial}}(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2),$$ (2.102)

$$k_{\text{exponential}}(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|),$$ (2.103)

$$k_{\text{tanh}}(x_i, x_j) = \tanh(\eta x_i \cdot x_j + c),$$ (2.104)

$$k_{\text{multiquadratic}}(x_i, x_j) = \sqrt{\|x_i - x_j\|^2 + c^2},$$ (2.105)

$$k_{\text{anova}}(x_i, x_j) = \sum_{p=1}^{n} \exp(-\sigma(x_i^p - x_j^p)^2)^d.$$ (2.106)

**Multiclass Problems**

Suppose now that we want to solve a multiclass problem (with the number of classes of more than two) with an SVM. The most popular approach to such tasks is called one-versus-all classification [Rifkin and Klautau, 2004]. In other words, we train a collective of binary classifiers that contains a separate model for each class. Each classifier learns to predict whether a data point belongs to the target class or to some other class.
2.4.2 K-Nearest Neighbours

The algorithm of k-Nearest Neighbours (k-NN) is a non-parametric method used for classification and regression. It is a representative of instance-based learning, or lazy learning, algorithms [Aha et al., 1991]. The algorithm was shown to be particularly effective for text classification [Masand et al., 1992; Khan et al., 2010]. k-NN computes how many times the data points of each class appear among the k points nearest to a new one the label of which needs to be predicted. As a result, the new data point is assigned to the majority class of its k nearest neighbours. The algorithm is illustrated in Figure 2.8 and can be formally described as follows. Suppose we have a training set S of n examples each of which consists of a p-dimensional feature vector \(x_i\) and a label \(y_i\):

\[
S = \{(x_1, y_1), \ldots, (x_n, y_n)\},
\]

\[
x_i \in \mathbb{R}^p,
\]

\[
y_i \in C,
\]

where C is a set of all classes and \(y_i\) indicates the class to which \(x_i\) belongs. Given some norm on \(\mathbb{R}^p\) and a new data point \(x^* \in \mathbb{R}^p\), we can reorder the set \(S\) in such a way that:

\[
S' = \{(x_{(1)}, y_{(1)}), \ldots, (x_{(n)}, y_{(n)})\},
\]

\[
||x_{(1)} - x^*|| \leq \ldots \leq ||x_{(n)} - x^*||.
\]

After that, the label \(y^*\) of the data point \(x^*\) can be determined as follows:

\[
y^* = \text{mode}(y_{(1)}, \ldots, y_{(k)}).
\]

![Fig. 2.8: Example of a simple k-NN-based classifier. The new data point depicted in black is assigned to the class of blue triangles if \(k = 2\). The same data point is assigned to the class of green squares if \(k = 6\).](image-url)
The parameter $k$ should be fine-tuned on a development set.

The following metrics can be used in order to compute distances in k-NN:

\[ d_{\text{euclidean}}(\mathbf{x}, \mathbf{x}^*) = \sqrt{\sum_{i=1}^{p} (x_i - x_i^*)^2}, \]  
\[ d_{\text{taxicab}}(\mathbf{x}, \mathbf{x}^*) = \sum_{i=1}^{p} |x_i - x_i^*|, \]  
\[ d_{\text{cosine}}(\mathbf{x}, \mathbf{x}^*) = 1 - \frac{\sum_{i=1}^{p} x_i x_i^*}{\sqrt{\sum_{i=1}^{p} x_i^2 \sum_{i=1}^{p} x_i^*^2}}. \]

**Weighted Voting**

In order to improve the classification results, we can apply a weighted voting procedure instead of just computing the mode in Formula 2.112. Weighted voting is particularly useful in case of imbalanced classes. The underlying idea is that the neighbours lying closer to the data point being classified make greater contributions to the decision of the model. For each class $c \in C$, we calculate the following:

\[ w(c,i) = \begin{cases} 
\frac{1}{d(\mathbf{x}(i), \mathbf{x}^*)} & \text{if } y(i) = c, \\
0 & \text{otherwise}. 
\end{cases} \]  
\[ w(c) = \frac{1}{k} \sum_{i=1}^{k} w(c,i), \]

where the function $d$ is a metric, e.g., one of those specified in Formula 2.113 – 2.115. Finally, the label $y^*$ of the data point $\mathbf{x}^*$ can be determined as follows:

\[ y^* = \arg\max_c (w(c)). \]

**2.4.3 Nearest Centroid Classifier**

Nearest Centroid Classifier, which is also known as Rocchio Classifier, is a simple classification algorithm proposed by Rocchio [1971]. The idea behind this classifier is that a new data point is assigned to the category the centroid of which lies closest to it. The centroid of a class can be interpreted as the centre of mass of all points belonging to this class given that each point has unit mass. This idea is illustrated in Figure 2.9. One centroid $g_c$ is computed for each class $c$ as follows:

\[ g_c = \frac{1}{|S_c|} \sum_{x \in S_c} x - \gamma \frac{1}{|S_{c,k}|} \sum_{x \in S_{c,k}} x, \]  

where $x$ is the feature vector of a training example, $S_c$ is the set of examples belonging to the class $c$, $S_{c,k}$ is the set of $k$ examples belonging to other classes and lying closest to the centroid $(1/|S_c|) \sum_{x \in S_c} x$, and the parameter $\gamma$ defines the importance of negative precedents. $\gamma$ is usually
chosen from the interval \([0, 1]\). The parameters \(k\) and \(\gamma\) together determine the tolerance of the algorithm to misclassification and should be fine-tuned on a development set. The metrics specified in Formula 2.113 – 2.115 can also be used to measure distances in the nearest centroid classifier.

Although the nearest centroid classifier was shown to be appropriate for text classification, SVM and k-NN tend to surpass it in terms of classification performance [Khan et al., 2010].

2.4.4 Artificial Neural Network

Artificial neural networks (ANNs) can be considered as computational systems simulating the processes that take place in a human’s brain. ANNs are often employed for modeling objects with completely unknown characteristics [Rutkovskaya et al., 2006]. Other typical applications of ANNs are image recognition, classification, and compression.

Likewise nervous tissue, any neural network consists of atomic units (neurons) that interact with one another using synapses [Tadeusiewicz, 1991]. A neuron receives signals via its incoming synapses that either make it active (and then it emits a high output signal) or keep it passive (and then its output signal remains low). This can be formally defined as follows:

\[
y = \begin{cases} 
1 & \text{if } \sum_{i=1}^{N} w_i u_i \geq v, \\
0 & \text{otherwise},
\end{cases}
\]  (2.120)

where \(u_1, \ldots, u_N\) are incoming signals, i.e., outputs of other neurons, \(w_1, \ldots, w_N\) are synaptic weights, \(y\) is the output of the current neuron, and \(v\) is a threshold value. This definition of a neuron can be rewritten as follows:
\begin{align*}
    y &= f \left( \sum_{i=0}^{N} w_i u_i \right), \\
    f(x) &= \begin{cases} 
        1 & \text{if } x \geq 0, \\
        0 & \text{otherwise},
    \end{cases}
\end{align*}
(2.121)

where \( w_0 = v \) and \( u_0 = 1 \) in addition to the previous definitions. This model of a neuron was proposed by McCulloch and Pitts [1943]. It is also possible to use other functions \( f \) besides the unit step function from Formula 2.122, e.g., the following ones:

\begin{align*}
    f(x) &= \begin{cases} 
        1 & \text{if } x \geq 0, \\
        -1 & \text{otherwise},
    \end{cases}
\end{align*}
(2.123)

\begin{align*}
    f(x) &= \begin{cases} 
        1 & \text{if } x > 1, \\
        -1 & \text{if } x < -1, \\
        x & \text{if } |x| \leq 1.
    \end{cases}
\end{align*}
(2.124)

\begin{equation}
    f(x) = \frac{1}{1 + e^{-\beta x}} > 0.
\end{equation}
(2.125)

\begin{equation}
    f(x) = \tanh \left( \frac{\alpha x}{2} \right) = \frac{1 - e^{-\alpha x}}{1 + e^{-\alpha x}} > 0.
\end{equation}
(2.126)

Interestingly, Formula 2.125, also known as sigmoid function, becomes Formula 2.122 when \( \beta \to \infty \). Similarly, Formula 2.126, also known as hyperbolic tangent, becomes Formula 2.123 when \( \alpha \to \infty \).

Connecting individual neurons with one another, we can construct complex architectures capable of solving challenging problems. Several possible ANN architectures are discussed in the present section below.

**Multilayer Feedforward Neural Network**

Multilayer feedforward fully connected ANN is one of the simplest neural architectures. Usually, the more layers of neurons a multilayer ANN has, the more abstract tasks it can solve. We illustrate the spreading of signals and errors in a multilayer fully connected ANN, using the Error Back Propagation algorithm (EBP), which is considered to be the basic method for training ANNs. An example of a multilayer fully connected ANN is depicted in Figure 2.10. This network consists of \( L \) layers. Each layer has \( N_k \) neurons, where \( k = 1, \ldots, L \). Each neuron is denoted as \( AD^k_i \), where \( i = 1, \ldots, N_k \). The network has \( N_0 \) inputs that receive signals \( u_1(n), \ldots, u_{N_0}(n) \) represented as vectors in the following form:

\[ u = [u_1(n), \ldots, u_{N_0}(n)]^T \]  
(2.127)

The output of the \( i \)-th neuron in the \( k \)-th layer is denoted as \( y^{(k)}_i(n) \), where \( i = 1, \ldots, N_k \) and \( k = 1, \ldots, L \).

The detailed architecture of an individual neuron \( AD^k_i \) is visualised in Figure 2.11. This neuron has \( N_{k-1} + 1 \) inputs that form the following vector:

\[ x^{(k)}(n) = [x_0^{(k)}(n), \ldots, x_{N_{k-1}}^{(k)}(n)]^T. \]  
(2.128)
The inputs of the neuron $AD^k_i$ are thus connected with the outputs of the previous layer $(k-1)$ in the following way:

$$x^{(k)}_j(n) = \begin{cases} 
    u_j(n) & \text{for } k = 1, \\
    y^{(k-1)}_j(n) & \text{for } k = 2, \ldots, L, \\
    1 & \text{for } j = 0 \text{ and } k = 1, \ldots, L.
\end{cases} \quad (2.129)$$

A value $w_{ij}^{(k)}(n)$ denotes the weight of the synapsis connecting the neuron $AD^k_i$, where $i = 1, \ldots, N_k$, with the input signal $x_j^{(k)}(n)$, where $j = 0, \ldots, N_{k-1}$. Let us denote the vector of the synaptic weights belonging to the neuron $AD^k_i$ as follows:

$$w_i^{(k)}(n) = [w_{i,0}^{(k)}(n), \ldots, w_{i,N_{k-1}}^{(k)}(n)]^T \text{ for } k = 1, \ldots, L \text{ and } i = 1, \ldots, N_k. \quad (2.130)$$
The output signal of the neuron \( AD_k^i \) for the \( n \)-th data example is computed in the following way:

\[
y^{(k)}(n) = f(s^{(k)}(n)),
\]

\[
s^{(k)}(n) = \sum_{j=0}^{N_{k-1}} w^{(k)}_{ij}(n)x^{(k)}_j(n).
\]

The outputs of the neurons in the \( L \)-th layer are as follows:

\[
y^{(L)}_1(n), y^{(L)}_2(n), \ldots, y^{(L)}_{N_L}(n).
\]

These signals are also the output signals of the ANN which are compared with targets:

\[
d^{(L)}_1(n), d^{(L)}_2(n), \ldots, d^{(L)}_{N_L}(n).
\]

As a result, we get the following output error:

\[
\varepsilon^{(L)}_i(n) = d^{(L)}_i(n) - y^{(L)}_i(n) \quad \text{for} \quad i = 1, \ldots, N_L.
\]

However, it is more convenient to minimise the squared error:

\[
Q(n) = \sum_{i=1}^{N_L} \left( \varepsilon^{(L)}_i(n) \right)^2 = \sum_{i=1}^{N_L} \left( d^{(L)}_i(n) - y^{(L)}_i(n) \right)^2.
\]

Given Formula 2.131 and 2.132, we note that Formula 2.136 is a function of synaptic weights. The learning process implies correcting the weights \( w^{(k)}_{ij}(n) \) in such a way that the function \( Q \) is minimised. For this purpose, we can use the gradient descent method:

\[
w^{(k)}_{ij}(n+1) = w^{(k)}_{ij}(n) - \eta \frac{\partial Q(n)}{\partial w^{(k)}_{ij}(n)},
\]

where the constant \( \eta > 0 \) determines a correction step. Given the chain rule and Formula 2.132, we also note that

\[
\frac{\partial Q(n)}{\partial w^{(k)}_{ij}(n)} = \frac{\partial Q(n)}{\partial s^{(k)}_i(n)} \frac{\partial s^{(k)}_i(n)}{\partial w^{(k)}_{ij}(n)} = \frac{\partial Q(n)}{\partial s^{(k)}_i(n)} x^{(k)}_j(n).
\]

If we denote

\[
\delta^{(k)}_i(n) = -\frac{1}{2} \frac{\partial Q(n)}{\partial s^{(k)}_i(n)},
\]

we will get the following:

\[
\frac{\partial Q(n)}{\partial w^{(k)}_{ij}(n)} = -2\delta^{(k)}_i(n)x^{(k)}_j(n).
\]

The algorithm defined by Formula 2.137 can thus be rewritten as follows:
\[
\begin{align*}
    w^{(k)}_{ij}(n+1) &= w^{(k)}_{ij}(n) + 2\eta \delta^{(k)}_i(n)x^{(k)}_j(n), \quad (2.141)
\end{align*}
\]

The way of computing the value \( \delta^{(k)}_i(n) \) defined by Formula 2.139 depends on a layer number. For the output layer, we obtain

\[
\begin{align*}
    \delta^{(L)}_i(n) &= -\frac{1}{2} \frac{\partial Q(n)}{\partial s^{(L)}_i(n)} = -\frac{1}{2} \frac{\partial}{\partial s^{(L)}_i(n)} \left( \sum_{m=1}^{N_L} \left( \varepsilon^{(L)}_m(n) \right)^2 \right) = -\frac{1}{2} \frac{\partial}{\partial s^{(L)}_i(n)} \left( \varepsilon^{(L)}_i(n) \right)^2 = e^{(L)}_i(n) \frac{\partial s^{(L)}_i(n)}{\partial s^{(L)}_i(n)} = e^{(L)}_i(n) f'(s^{(L)}_i(n)). \quad (2.142)
\end{align*}
\]

For an arbitrary layer \( k \neq L \), we obtain

\[
\begin{align*}
    \delta^{(k)}_i(n) &= -\frac{1}{2} \frac{\partial Q(n)}{\partial s^{(k)}_i(n)} = -\frac{1}{2} \sum_{m=1}^{N_{k+1}} \left[ \frac{\partial Q(n)}{\partial s^{(k+1)}_m(n)} \frac{\partial s^{(k+1)}_m(n)}{\partial s^{(k)}_i(n)} \right] = \\
    &= \sum_{m=1}^{N_{k+1}} \left[ \delta^{(k+1)}_m(n)w^{(k+1)}_{mi}(n) f'(s^{(k)}_i(n)) \right] = f'(s^{(k)}_i(n)) \sum_{m=1}^{N_{k+1}} \left[ \delta^{(k+1)}_m(n)w^{(k+1)}_{mi}(n) \right]. \quad (2.143)
\end{align*}
\]

Let us define the error in the \( k \)-th layer \( (k \neq L) \) for the \( i \)-th neuron as follows:

\[
\begin{align*}
    \varepsilon^{(k)}_i(n) &= \sum_{m=1}^{N_{k+1}} \left[ \delta^{(k+1)}_m(n)w^{(k+1)}_{mi}(n) \right] \quad \text{for } k = 1, \ldots, L - 1. \quad (2.144)
\end{align*}
\]

Substituting Formula 2.144 into Formula 2.143, we obtain

\[
\begin{align*}
    \delta^{(k)}_i(n) &= \varepsilon^{(k)}_i(n)f'(s^{(k)}_i(n)). \quad (2.145)
\end{align*}
\]

As a result, the EBP algorithm can be written as follows:

\[
\begin{align*}
    s^{(k)}_i(n) &= \sum_{j=0}^{N_k} \left[ w^{(k)}_{ij}(n)x^{(k)}_j(n) \right], \quad (2.146) \\
    y^{(k)}_i(n) &= f(s^{(k)}_i(n)), \quad (2.147) \\
    \varepsilon^{(k)}_i(n) &= \begin{cases} 
        d^{(L)}_i(n) - y^{(L)}_i(n) & \text{for } k = L, \\
        \sum_{m=1}^{N_{k+1}} \left[ \delta^{(k+1)}_m(n)w^{(k+1)}_{mi}(n) \right] & \text{for } k = 1, \ldots, L - 1, 
    \end{cases} \quad (2.148) \\
    \delta^{(k)}_i(n) &= \varepsilon^{(k)}_i(n)f'(s^{(k)}_i(n)), \quad (2.149) \\
    w^{(k)}_{ij}(n+1) &= w^{(k)}_{ij}(n) + 2\eta \delta^{(k)}_i(n)x^{(k)}_j(n). \quad (2.150)
\end{align*}
\]

The name of the algorithm is connected with the way how we compute errors in individual layers. Signals spread through the ANN forward starting from the first layer, while errors propagate backward starting from the last layer. The initial values of the synaptic weights should be small enough; they are usually generated from a random distribution with a mean value equal to 0.
Prior to the training process, the entire data set is usually divided into mini-batches if it is big enough. Next, the gradient is summed over each mini-batch, and model updates are performed once for each mini-batch as well. The batchwise training thus reduces computational costs and the variance of the gradient. One epoch is over when we have fed all the training examples (elementwise or batchwise) to our model. The parameter $\eta$, also known as learning rate, should be great enough at the first epochs for faster convergence. Afterwards, the learning rate should be reduced for better fine-tuning.

There exist different modifications of the EBP algorithm [Cochocki and Unbehauen, 1993]. One of them introduces an additional item (momentum) into Formula 2.150:

$$w_{ij}^{(k)}(n+1) = w_{ij}^{(k)}(n) + 2\eta\delta_i^{(k)}(n)x_j^{(k)}(n) + \alpha \left[ w_{ij}^{(k)}(n) - w_{ij}^{(k)}(n-1) \right],$$

(2.151)

where $\alpha \in (0, 1)$. The study [Tadeusiewicz, 1993] demonstrates that introducing a momentum accelerates the algorithm indeed.

There also exist more modern methods, e.g., [Tieleman and Hinton, 2012; Karayiannis and Venetsanopoulos, 2013; Kingma and Ba, 2014], which train ANNs faster (with fewer iterations) than the classical EBP algorithm does. However, most of them are based on stochastic gradient descent just like the EBP method described above.

Multilayer ANNs are successfully applied to various problems, e.g., text classification [Sergienko et al., 2015] and lexical H-M AD [S. V. Ravuri and Stolcke, 2014].

Convolutional Neural Network (CNN) is an ANN architecture first applied by LeCun et al. [1998] to image recognition. However, the idea behind CNNs was proposed even earlier by Fukushima and Miyake [1982]. The architecture of a CNN is analogous to the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. In particular, the following analogies are used in CNNs. First, there exist two types of neurons discovered in the visual cortex: simple neurons that react to straight lines at specific angles, and complex neurons that react activating specific sets of simple neurons. Second, individual neurons respond to stimuli only in a restricted region of the visual field. This region is known as the receptive field. A collection of such overlapping fields covers the entire visual area.

A CNN is able to assign importance (learnable weights and biases) to various aspects/objects in an input image and also to distinguish one from the other. The most important advantage of CNNs is a significantly smaller amount of input data preprocessing required in comparison with other ML algorithms. Primitive methods usually employ hand-engineered filters (feature extractors), while CNNs are capable of learning their own feature representations which are optimal for a specific task. Another important advantage of CNNs is a significantly smaller number of learnable parameters in comparison with a simple multilayer fully connected ANN (with a flattened input image) required to solve the same task. The reason for that is the relatively small size of a kernel, the weights of which can be reused throughout an image. The architecture of a CNN is thus less redundant and better regularised than a multilayer fully connected ANN.

Let us consider an input image with a resolution of $a \times b \times c$ pixels, where $a$ and $b$ are the height and width of the image respectively, and $c$ is the number of channels that equals 3 for RGB images.
and 1 for greyscale images. For a convolution, we need to define a kernel size of \( k \times k \) pixels (square kernel). The parameter \( k \) is usually chosen to be \( k << a \) and \( k << b \) for the first convolutional layer. For each channel, several convolutional kernels (filters) are randomly initialised with weights close to 0. Each kernel shifts from left to right through the entire image of its channel at a stride of \( s \) pixels \((s \leq k)\) until it reaches the right border of the image. After that, the kernel returns to the left border of the image, shifts one stride down, and the process repeats until the entire image is covered. In each position of the kernel, the elements of the kernel are multiplied elementwise by the elements of the image region currently covered by the kernel. The results of this operation are then summarised within the current image region and the sum is saved to the respective element of a matrix. This matrix, obtained after shifting the kernel all over the image, is called a feature map.

Prior to a convolution, it is sometimes useful to pad the initial input image with zeros around it. Next, an activation function is applied to each element of the feature map. We can use any of the activation functions defined in Formula 2.124 – 2.126. However, there exists a specific function that was shown to be particularly efficient for large CNNs in the sense of computational complexity. This function is called Rectified Linear Unit (ReLU) and defined as follows:

\[
f(x) = \max(0, x).
\]  

(2.152)

Next, the activated feature map, which can also be considered as one channel of the convolved image, is subjected to downsampling. This procedure is also known as pooling. Similarly to the convolution, a pooling operation is performed within a pooling kernel of \( p \times p \) pixels that shifts at a stride of \( l \) pixels \((l \leq p)\) all over the activated feature map. The parameter \( p \) is usually chosen to be \( p << a \) and \( p << b \) for the first pooling layer. There exist several types of pooling, e.g., average pooling and max pooling. The former pooling type returns the average of the elements within the activated feature map region currently covered by the pooling kernel. The latter pooling type returns the maximum of these elements.

A convolution, an activation, and a pooling operator are usually stacked in this order. As a rule, a convolution and an activation operator are employed together and form a convolutional layer, while the pooling operator after them may be missing. The number of layers, the number of filters in each layer, and the kernel sizes should be carefully fine-tuned for a specific task since these hyper-parameters influence the characteristics of features extracted by the CNN. Each new convolutional layer returns more abstract features, e.g., the first convolutional layer detects straight lines at specific angles, the second one recognises simple geometric figures, the next one detects complex objects, etc.

Solving a classification task with a CNN, we flatten the output of its last layer and append one or more fully connected layers. The last fully connected layer contains \( K \) neurons, where \( K \) is the number of classes. Finally, we can apply the softmax function \( s \) to the weighted sum \( z_i \) of the input signals for the \( i \)-th neuron in the latter layer. This function is defined as follows:

\[
s(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for} \quad i = 1, \ldots, K \quad \text{and} \quad z = (z_1, \ldots, z_K) \in \mathbb{R}^K.
\]  

(2.153)

Instead of softmax, we can also employ other activation functions for the neurons of the last fully connected layer, e.g., those specified in Formula 2.124 – 2.126. If necessary, we can rescale the
activation function result to be in the range of $[0, 1]$ in case our classification targets are represented by one-hot vectors. In such a vector, only one component equals 1 and denotes the correct class, while the other components are zeros.

An example of a CNN is visualised in Figure 2.12. This neural network performs two-dimensional convolutions of greyscale input images to classify handwritten digits. CNNs can also be applied to audio analysis, e.g., when a CNN carries out one-dimensional convolutions of time-continuous raw audio signals [Trigeorgis et al., 2016; Schuller et al., 2017], or when a CNN performs two-dimensional convolutions of time-continuous audio spectrograms.

Recurrence Neural Network

Recurrent Neural Networks (RNNs) are based on [Rumelhart et al., 1986]. It is a class of ANNs that allow previous outputs to be used as inputs while having hidden states. Therefore, such networks are extremely useful for processing time series, e.g., speech. An example of an RNN is illustrated in Figure 2.13a. For each time step $t$ and input $x_t$, the hidden state $h_t$ and the output $y_t$ are computed as follows:

$$h_t = g_1(W_h h_{t-1} + W_x x_t + b_h),$$  \hspace{1cm} (2.154)

$$y_t = g_2(W_y h_t + b_y),$$  \hspace{1cm} (2.155)

where $W_h$, $W_x$, and $W_y$ are learnable weights and $b_h$ and $b_y$ are biases. All of these parameters are shared temporary. $g_1$ and $g_2$ are activation functions, e.g., tanh. For RNNs, a loss function $L$ of all time steps is computed based on the loss at every time step:
2.4 Classification Models

\[ L(\hat{y}, y) = \sum_{t=1}^{T} L(\hat{y}_t, y_t), \quad (2.156) \]

where \( \hat{y} \) and \( y \) are the target output sequence and the actual output sequence respectively. Error back propagation is also performed at each point in time. At the time step \( T \), the derivative of the loss function \( L \) with respect to the weight matrix \( W \) is defined as follows:

\[ \frac{\partial L^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial L^{(T)}}{\partial W} \bigg|_{(t)}. \quad (2.157) \]

Such an architecture endows RNNs with the following advantages over feedforward ANNs:

- Possibility to process input of any length;
- Weights being shared across time;
- Model size not increasing with the size of input;
- Computation taking into account historical information.

However, RNNs also have significant drawbacks:

- Relatively slow computation;
- Inability to consider any future input for the current state;
- Difficulty of accessing information from a long time ago.

The latter issue is called the problem of long-term dependences [J. Hochreiter, 1991; Bengio et al., 1994] and can be illustrated as follows. Let us compare two phrases: "The birds are flying in the sky." and "I grew up in Russia. ... I speak fluent Russian." In both cases, we need to predict the word \( y_t \) in italic given the previous context \( x_1, \ldots, x_{t-1} \). An RNN will succeed in predicting the word "sky", while it will be likely to fail predicting the word "Russian". The reason for that is the relevant context located too far from the word being predicted in the latter phrase. In a more technical sense, it is difficult to capture long-term dependencies because of multiplicative gradient that can be exponentially vanishing or exploding with respect to the number of layers.

Long Short-Term Memory Networks (LSTMs) are a special kind of RNNs, introduced by S. Hochreiter and Schmidhuber [1997]. LSTMs were specially designed to resolve the problem of long-term dependencies. Both a simple RNN and an LSTM architecture are visualised in Figure 2.13. The key feature of any LSTM network is a cell state (see the horizontal line running through the top of Figure 2.13b). The cell state stores important contextual information that flows through the entire chain with only minor changes at each new time step \( t \). The LSTM is able to remove old, useless information from the cell state and also to add new, important information to the cell state by means of gates. Let us now walk step by step through the LSTM from Figure 2.13b to show how these gates work.

At the first step, the LSTM decides which old information should be removed from the cell state. This decision is made by a sigmoid layer called the "forget gate layer" and specified as follows:

\[ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \quad (2.158) \]

where \( \sigma \) is the sigmoid function applied to its arguments elementwise, \( W_f \) and \( U_f \) are matrices of learnable weights for the input connections and for the recurrent connections respectively, \( b_f \)
is a vector of biases, $x_t$ and $h_{t-1}$ are the current input vector and the previous hidden state vector respectively, and $t$ is a time step index. An $f_t$ component value close to 0 means forgetting the corresponding component of the cell state vector $C_{t-1}$, while an $f_t$ component value close to 1 stands for preserving it. Each gate of the LSTM has its own matrices $W$ and $U$ and a vector $b$.

(a) Simple RNN architecture. The $\rightarrow$ sign inside a circle denotes the elementwise application of the activation function the outgoing arrow points to. $W_x$, $W_h$, and $W_y$ are learnable weights, $b_h$ and $b_y$ are biases, and $g_1$ and $g_2$ are some activation functions.

(b) LSTM architecture. The rectangles are trainable layers with their own weights and biases (not depicted in the figure). $\sigma$ is the sigmoid activation function. $\rho$ is also some activation function that is optional depending on what values we need to output at each time step.

Fig. 2.13: Comparison of RNN and LSTM. Each line carries an entire vector from the output of one node to the inputs of the others. The circles represent elementwise operations. A split line means its content being copied and the copies going to different destinations.
At the second step, the LSTM determines which new information should be added to the cell state. First, a sigmoid layer called the “input gate layer” decides which values will be updated:

\[ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i). \]  
(2.159)

Next, a tanh layer returns a vector of new candidate values \( C^*_t \) that could be added to the cell state:

\[ C^*_t = \tanh(W_C x_t + U_C h_{t-1} + b_C), \]  
(2.160)

where \( \tanh \) is the hyperbolic tangent function applied to its arguments elementwise.

At the third step, the LSTM updates the cell state \( C_{t-1} \) with the information obtained at the previous two steps:

\[ C_t = f_t \circ C_{t-1} + i_t \circ C^*_t, \]  
(2.161)

where \( \circ \) denotes Hadamard (elementwise) product.

At the fourth step, the LSTM computes a new hidden state based on the updated cell state \( C_t \), the current input \( x_t \), and the previous hidden state \( h_{t-1} \):

\[ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \]  
(2.162)

\[ h_t = o_t \circ \tanh(C_t). \]  
(2.163)

Sometimes it is also useful to add another layer to modify the hidden state for the subsequent output, if we want our LSTM to produce some value \( y_t \) at each time step:

\[ y_t = \rho(W_y h_t + b_y), \]  
(2.164)

where \( \rho \) is an activation function.

Deep Neural Network

Deep Neural Network (DNN) is quite a subjective term that denotes an ANN with multiple hidden layers. Although there exists no exact threshold for the number of layers or the number of learnable parameters that an ANN should have to be considered as a DNN, some specific architectures belong to DNNs by default regardless of their sizes, e.g., CNN, RNN, and LSTM. The main reason for that is the theoretical possibility to split the data representations inside the model into various abstraction levels.

Despite their impressive perceptive abilities, DNNs are usually tricky to configure and train that makes their applications to many real-life tasks difficult due to a lack of training data.

DNNs have already been employed for some applications related to AD. S. Ravuri and Stolcke [2015], Ouchi and Tsuboi [2016], and R. Zhang et al. [2018] used RNNs in their studies on text-based AD. Their models were designed to work with distributed word representations. Mallidi et al. [2018] and C.-W. Huang et al. [2019] applied LSTMs to audio-based AD. Schuller et al. [2017] and Norouzian et al. [2019] stacked a CNN and an LSTM network into a single model designed for audio-based AD. The model proposed by Schuller et al. [2017] was receiving raw audio and
performing a series of one-dimensional convolution and pooling operations across time, while the model designed by Norouzian et al. [2019] was working with spectrograms and performing a series of two-dimensional convolution and pooling operations across frequency and time.

### 2.5 Fusion

Fusion enables leveraging data from different sources including data sets, feature sets, or modalities in order to achieve more accurate predictions. There exists a great variety of fusion techniques depending on the nature of data sources, model architectures, abstraction levels at which the fusion is performed, etc. [Castanedo, 2013]. In the present study, we consider the following classification of fusion techniques based on the latter criterion:

- Data-level fusion;
- Feature-level fusion;
- Model-level fusion;
- Decision-level fusion;
- Complex fusion.

These fusion types are discussed in detail in the present section below. The usage examples given for the fusion types being considered are not direct instructions on how fusion should be used for a particular task. In real applications, the optimal fusion type should usually be determined empirically. It is also possible to combine different fusion types inside a single model.

#### 2.5.1 Data-Level Fusion

This is the simplest type of fusion that is appropriate in case our data sets fully complement one another, e.g., when we merge recordings of several English speakers in order to train an ASR system. An example of the data-level fusion pipeline is depicted in Figure 2.14a.

#### 2.5.2 Feature-Level Fusion

This type of fusion can be used in case the data sets are of similar but different nature and contain the same set of observations, e.g., when we want to determine the position of an object by using two cameras placed at different angles. An example of the feature-level fusion pipeline is illustrated in Figure 2.14b. The model receiving concatenated features can now determine correlated attributes on its own. The number of the features being concatenated should be small enough to minimise the negative effect of the dimensionality curse.

#### 2.5.3 Model-Level Fusion

In case the extracted feature sets are too large to be fused at the feature level, it is possible to apply model-level fusion as illustrated in Figure 2.14c. Let us take an ANN as an example. First, Network 1 on the left side and Network 2 on the right side of the figure are trained separately each on its own modality. Second, the last layer of each network is cut off and the outputs of the preceding layers
of both models are fed to Layer 3 in the middle of the figure. After that, this layer is trained on the fused data out of scratch, while the parameters of the other layers can either be fixed or fine-tuned with a small learning rate.

#### 2.5.4 Decision-Level Fusion

In case our modalities are of completely different nature, it is useful to apply decision-level fusion as illustrated in Figure 2.14d. The predictions obtained from several models each of which deals with its own modality are concatenated into a vector of metafeatures that is fed to a metamodel returning the final prediction. Sometimes the metamodel can be replaced by a simple postprocessing stage,
e.g., the predictions obtained from the low-level models for a classification task can be averaged to get the final prediction.

### 2.5.5 Complex Fusion

Transfer learning illustrated in Figure 2.14e is an example of complex fusion. Transfer learning is an ML technique where a model trained on one task is repurposed on another related task [Goodfellow et al., 2016]. Suppose we want to determine the situations when a human looks directly at the camera. We have two data sets at our disposal: a large data set of various objects including human faces in arbitrary positions and a much smaller data set of human faces including the frontal faces that we want to detect. First, Network 1 depicted on the left side of Figure 2.14e is trained on the large data set to distinguish human faces from other objects. This model provides abstract numerical representations of human faces as such. Second, we cut off the last layer of Network 1 and transfer the remaining elements of the model to a new model depicted on the right side of the same figure. The new model is trained on the small data set to distinguish frontal faces from the rest. The last layer of Network 2 is trained out of scratch, while the parameters of the other layers can either be fixed or fine-tuned with a small learning rate.

### 2.6 Summary

In the present chapter, we have discussed several existing studies on AD. We have also described various feature sets and several ML approaches which can be applied to AD. In our thesis, we address the following issues given the related work from Section 2.1.

The first issue is connected with data. Our data sets are supposed to contain realistic, spontaneous conversations modelling different scenarios of AD. The data sets should be appropriate for both training and testing our prosed classifiers. It is also interesting to investigate AD for other languages besides English since most of the existing studies were dedicated to this language exclusively.

The second issue is connected with multimodality. The existing studies confirm that leveraging as many data modalities as possible can improve the AD performance. Therefore, at least acoustic and ASR-based features should be involved in the process of AD. It is also desirable to have other data modalities besides speech and text at our disposal.

The third issue is connected with the ML methodology. The use of DNNs in combination with low-level features can improve the AD performance compared to the traditional models. In order to combat the lack of training data for DNNs, we utilise several corpora and introduce a novel approach to data augmentation based on the mixup method [H. Zhang et al., 2017]. Alongside with complex DNN-based models, it is always useful to implement simpler models in combination with more abstract features first. As a rule, simpler models are easier to configure and train. Therefore, we apply both simple and complex classifiers in our experiments for speech and text analysis on AD.
Corpora and Tools

In this chapter, we describe four corpora and several tools which we use to conduct our experiments. We have chosen these corpora to resolve the data and multimodality issues which we pointed out in Section 2.6. The present chapter is organised as follows: the SmartWeb Video Corpus (SVC), the Voice Assistant Conversation Corpus (VACC), the Restaurant Booking Corpus (RBC), and the HomeBank Child/Adult Addressee Corpus (HB) are considered in Section 3.1, 3.2, 3.3, and 3.4 respectively. Section 3.5 is dedicated to tools and computational infrastructure. The content of the present chapter is summarised in Section 3.6.

3.1 SmartWeb Video Corpus

Two data sets containing utterances in the context of a visit to some event of the Football World Cup 2006 have been recorded within the SmartWeb project [Wahlster, 2004]. Different categories of Off-Talk were induced (in the SW$_{spont}$ data set) or acted (in the SW$_{acted}$ data set) [Batliner, Hacker, and Nöth, 2008]. In the present study, we consider only the SW$_{spont}$ data set, which is also known as the SmartWeb Video Corpus (SVC). Within the SmartWeb project, a mobile multimodal user interface has been developed for the Semantic Web system. Using this interface, people can ask open-domain questions to the system, no matter where they are; carrying a smartphone, they address the system through speech and eye gaze [Reithinger et al., 2005]. The idea of SVC is to classify automatically whether the user’s speech is addressed to the system (On-Talk) or to someone else (Off-Talk). The system is supposed to perform automatic AD without a PTT button so that the dialogue manager does not get confused. SVC thus represents the classical H-M AD problem (see Section 1.2.1 for more detail).

3.1.1 Experimental Setup

SVC contains the user’s speech and visual data collected from the user’s head and face during a simulated dialogue. The audio-visual information was recorded with a close talk microphone and a video camera embedded into the frontal panel near the display of a mobile platform. The purpose of the corpus was to enable a module of the SmartWeb system to detect the user’s focus of attention based on this multimodal data. The idea behind using the audio modality is that people tend to modify their normal manner of speech and prosody as soon as they start talking to the SDS. The
underlying idea of using the video modality is that users tend to look at the system display while interacting with the SDS in the absence of distracting factors. They do this due to the necessity to receive the visual feedback that appears on the system display, e.g., the n best results of a query, maps, pictures, or videos. Two audio tracks and one video track were recorded within each session. Recording locations were selected among real-life situations with acoustic and visual noise of varying degree. The tracks were recorded with resolutions and sample rates that could be expected from a scenario like a question answering system using mobile networks of that time.

The video track without embedded audio was recorded with a fixed focus camera placed in the frontal panel of a Nokia 6680 cell phone that was carried by the participants. The user could observe the view of the camera on the display, and therefore the majority of the video tracks show the user’s head roughly centred. The video was coded with H.263 directly on the cell phone. The first audio signal was recorded with a headset microphone and transmitted over a Bluetooth connection to the cell phone and afterwards via Wideband Code Division Multiple Access (WCDMA) to a voice processing server located at the Bavarian Archive for Speech Signals in Munich (BAS). Due to the chosen transmission technique, this signal was recorded using the 8-bit resolution and a sample rate of 8000 Hz with the A-law coding following the protocol specification ITU-T G.711. The second audio signal was received by a simple collar mounted microphone at 44100 Hz with the 16-bit Pulse Code Modulation (PCM) coding.

During the whole session, the cell phone was connected to the voice processing server. The server was capable of recording speech, detecting the end of speech by means of silence, and emitting arbitrary audio files. The sequence of recordings and acoustic prompts was determined by an XML-file generated randomly for each experimental session. The prompts had been prerecorded with a synthesised voice.

3.1.2 Data Collection

The data collection design was aimed at gathering multimodal data in mobile, realistic environments in order to detect the Off-Talk phenomenon automatically. For dialogue simulation, the Situational Prompting technique (SitPro) [Mögele et al., 2006; Kaiser et al., 2006] was applied. In contrast to the ordinary Wizard-of-Oz (WOZ) simulation method, system reactions are scripted within a SitPro experiment and users know that the system is simulated. The SitPro approach integrates script methods with interview techniques and speaker prompting [Gibbon et al., 1997; Rapp and Strube, 2002] into so called standard prompts, individualised prompts, and scripted prompts. In a standard prompt, the participant is told a characteristic topic of a subject area, e.g., soccer (team, group), navigation (public transport, pedestrians), community (restaurant, night club), or information (tourist information, points of interest) to which he/she needs to pose a query. An individualised prompt is a prompt for which the participant provides his/her own topic. A scripted prompt simulates a three-turn conversation as frequently exhibited in H-M dialogues. For this purpose, an instructor with a female voice and an operator with a male voice simulate the SDS. A system prompt (instructor) followed by a variable silence interval, the participant’s utterance (user), and a possible system answer (operator) form a prompt unit. Six prompt units are bundled into an action unit as a single thematic episode. In order to collect a lot of realistic Off-Talk utterances and head movements of the participant, SitPro was used in a scenario with two human subjects (a user and a companion) and the simulated SDS on a mobile device as shown in Figure 3.1.
The recordings took place in various indoor and outdoor locations, e.g., an office, a coffee bar, or a park. Two human subjects participated in each experiment. The speech and video data of the user were recorded. His/her tasks were to interact with the SDS for requesting information on various topics of interest from it, to read the output on the system display aloud, to retell the information in arbitrary verbal form to the second subject (companion), and to answer the companion’s questions. The instructor of the SDS was giving directions about the task, the situation, and the topics. The operator was "answering" the user’s questions or giving feedback similar to the target SmartWeb system. Prior to the experiment, the companion received a note with nine listed possibilities to distract the user from the call task. Distracting phrases may be as follows: "Ask your partner to get information about sightseeing nearby.", “Tell your partner to hurry up.", or “Ask your partner something concerning his/her most recent question.”. The companion could also invent his/her own interposing phrases on the spot. As a result, a controlled triadic communication scenario was designed. In this scenario, the companion could observe the user only; neither system speech nor display output were available to the companion. Furthermore, the companion has never talked to the SDS directly.

To induce Spontaneous Off-Talk (SOT), the companion interfered in the interaction between the user and the system with interposing phrases. In order to elicit Paraphrased Off-Talk (POT), the user was asked to report to the companion what information he/she had found interacting with the system. These two actions provoke changing from On-View (the user is looking directly into the camera) to Off-View (the user is looking somewhere else) and back since the user had to move his/her head in the direction of the companion to react to his/her interruptions. Read Off-Talk (ROT) occurs when the user reads the text on the system display aloud. Off-Talk utterances that do not fit into these three categories, e.g., thinking aloud or swearing, were labelled as Other Off-Talk (OOT). On-Talk utterances are also denoted as Not Off-Talk (NOT). After a ten-minute recording, the participants of the experiment swapped their roles. No specific requirements regarding Off-Talk were given to the participants in order to obtain as realistic H-H-M conversations as possible. Examples of conversational fragments and their translations from German into English are given in Table 3.1 – 3.3.
3.1.3 Data Annotation

The corpus contains 99 dialogues (each with a unique user) including 2193 utterances (25073 words) with a total duration of 3 hours and 28 minutes\(^1\). Each word in the corpus has been manually annotated. The label distribution is shown in Table 3.4. In all experiments, OOT was mapped onto SOT. For each utterance, the utterance label was calculated by a majority voting among all the word labels in the utterance. This word-to-utterance label transformation is described in [Noth et al., 2007].

The annotation of the video recordings includes the segmentation of faces with a surrounding rectangle as well as the manual frame-based labelling (7.5 frames per second (FPS)) of the following four classes: On-View (79%), between On- and Off-View (5%), Off-View (14%), and No-Face (2% of the entire corpus duration). No-Face means that the user is out of view of the camera. The segmentation of faces was performed with the automatic face detector of the OpenCV library and using the manual segmentation of the On-View frames where the detector failed. In case of On-View, both eyes and the nose are present in the image but can be partially obscured, e.g., with a hand. A couple of video frames illustrating On-View and Off-View is given in Figure 2.4.

Table 3.5 summarises the corpus characteristics\(^1\). More detailed information on the SmartWeb project and SVC can be found in [Batliner et al., 2007; Batliner, Hacker, and Nöth, 2008].

Table 3.1: Example of ROT (italic). Original German utterances are given in a different font.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructor → User</td>
<td>You would like to know how long public transportation is available at night.</td>
</tr>
<tr>
<td></td>
<td>Du möchtest wissen, wie lange die öffentlichen Verkehrsmittel nachts verfügbar sind.</td>
</tr>
<tr>
<td>User → Operator</td>
<td>How long is the underground running during the night?</td>
</tr>
<tr>
<td></td>
<td>Wie lange läuft die U-Bahn in der Nacht?</td>
</tr>
<tr>
<td>Instructor → User</td>
<td>SmartWeb is going to display the answer. You can find the answer on card number eight. Please read it out aloud.</td>
</tr>
<tr>
<td>User → Companion</td>
<td>Regular lines are running until 2 A.M.. Afterwards there are night lines.</td>
</tr>
<tr>
<td></td>
<td>Die regulären Linien fahren bis 2 Uhr. Danach verkehren die Nachlinien.</td>
</tr>
</tbody>
</table>

\(^1\) The metadata concern the corpus version available from the catalogue of the European Language Resources Association as of 01.12.2015. The video tracks of 4 out of the 99 dialogues are damaged and can not be used for visual AD. This corpus version is utilised in the present study.
Table 3.2: Example of POT (italic). Original German utterances are given in a different font.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Utterance</th>
</tr>
</thead>
</table>
| Instructor → User | Now please ask the system how many times Bayern-Muenchen has already beaten the Hamburger SV.  
Fragen Sie jetzt das System, bitte, wie oft die Bayern-München die Hamburger SV bereits besiegt hat. |
| User → Operator | How many times has Bayern-Muenchen already beaten the HSV?  
Wie oft hat die Bayern-München die HSV schon besiegt? |
| Operator → User | Up to now Bayern-Muenchen has beaten the Hamburger SV four times.  
Bisher hat die Bayern-München die Hamburger SV vier mal besiegt. |
| User → Companion | Bayem-Muenchen has already beaten the HSV four times.  
Bayern-München hat die HSV schon vier mal besiegt. |

Table 3.3: Example of SOT (italic). Original German utterances are given in a different font.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Utterance</th>
</tr>
</thead>
</table>
| Companion → User | Do we actually have a ticket for the underground?  
Haben wir eigentlich eine Fahrkarte für die U-Bahn? |
| User → Companion | Yes, I bought a ticket.  
Ja, ich habe eine Fahrkarte gekauft. |
| Companion → User | Great. That’s good to know.  
Perfekt. Das ist gut zu wissen. |

Table 3.4: Label distribution as percentages of the total number of items in SVC.

<table>
<thead>
<tr>
<th></th>
<th>NOT</th>
<th>SOT</th>
<th>POT</th>
<th>ROT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word level</td>
<td>46.1</td>
<td>22.4</td>
<td>18.6</td>
<td>12.9</td>
</tr>
<tr>
<td>Utterance level</td>
<td>49.6</td>
<td>21.6</td>
<td>14.7</td>
<td>14.1</td>
</tr>
</tbody>
</table>
Table 3.5: General characteristics of SVC.

<table>
<thead>
<tr>
<th>SDS type</th>
<th>Portable SDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modalities</strong></td>
<td>Audio; Text (manual transcripts); Video</td>
</tr>
<tr>
<td><strong>Addressee labels (audio &amp; text)</strong></td>
<td>NOT (machine-addressed); SOT (human-addressed); POT (human-addressed); ROT (human-addressed)</td>
</tr>
<tr>
<td><strong>Visual focus labels (video)</strong></td>
<td>On-View; Between On- and Off-View; Off-View; No-Face</td>
</tr>
<tr>
<td><strong>SDS behaviour</strong></td>
<td>WOZ</td>
</tr>
<tr>
<td><strong>Speakers’ behaviour</strong></td>
<td>real (non-acted)</td>
</tr>
<tr>
<td><strong>Speakers/sessions</strong></td>
<td>99</td>
</tr>
<tr>
<td><strong>Gender</strong></td>
<td>Male: 36; Female: 63</td>
</tr>
<tr>
<td><strong>Total recorded data</strong></td>
<td>3 h 28 min, fully annotated</td>
</tr>
<tr>
<td><strong>Age</strong></td>
<td>Mean: 26.21; Std. dev.: 7.70; Min: 15; Max: 64</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>German</td>
</tr>
<tr>
<td><strong>Annotation</strong></td>
<td>Utterance segmentation; Addressee; Visual focus; Transcription; Native dialect; Location; Smoker; Glasses; Beard; Bald head; Piercing</td>
</tr>
<tr>
<td><strong>Native dialect / native language (if not German)</strong></td>
<td>Bayern: 60; Baden-Wuerttemberg: 12; Nordrhein-Westfalen: 4; Hessen: 4; Thueringen: 2; Rheinland-Pfalz: 2; Schleswig-Holstein: 2; Brandenburg: 1; Mecklenburg-Vorpommern: 1; Saarland: 1; Sachsen: 1; Sachsen-Anhalt: 1; Swiss German: 1; Romanian: 3; Russian: 1; Persian: 1; Polish: 1; Slovak: 1</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Indoor: 72; Outdoor: 27</td>
</tr>
<tr>
<td><strong>Smoker</strong></td>
<td>Yes: 32; No: 67</td>
</tr>
<tr>
<td><strong>Glasses</strong></td>
<td>Yes: 31; Sunglasses: 2; No: 66</td>
</tr>
<tr>
<td><strong>Beard</strong></td>
<td>Chin beard: 1; Moustache: 2; Chin beard &amp; moustache: 3; Full beard: 3; No: 90</td>
</tr>
<tr>
<td><strong>Bald head</strong></td>
<td>Yes: 3; No: 96</td>
</tr>
<tr>
<td><strong>Piercing</strong></td>
<td>Ear piercings: 35; Ear piercings &amp; eyebrow piercing: 1; Nose piercing: 2; No: 61</td>
</tr>
</tbody>
</table>
3.2 Voice Assistant Conversation Corpus

Similarly to SVC, Voice Assistant Conversation Corpus (VACC) represents the classical H-M AD problem (see Section 1.2.1 for more detail). Creating VACC, Siegert, Krüger, et al. [2018] were motivated by the following issues:

- How do users change their manner of speech when they talk to a modern SDS and switch to a human interlocutor?
- Which differences in the user’s speaking style during the interaction with the technical system can be observed when they are alone or with a companion?
- Do the users themselves recognise differences in their speaking style when they change their addressees (the system or the companion)?
- Do the observed and/or reported changes in their interaction style differ between a formal and an informal interaction setting?

3.2.1 Corpus Design

VACC consists of recordings capturing H-H-M interaction experiments. Each experiment was accompanied by various questionnaires presented before and after the experiment. The participants received initial instructions on the basic capabilities and the trigger-word-based addressing mechanism of Amazon Alexa. Each experiment included two modules (Calendar Module and Quiz Module) having different interaction complexity levels. The order of these modules was fixed for all experiments. Two conditions permuted for different participants were applied to each module. Under the first condition, the participant was solving the task alone. Under the second condition, the participant was solving the task together with a companion. In other words, each experiment consisted of four “rounds”. One round was finished when the interaction aim was reached, or interrupted to avoid confusion if any success could hardly be achieved.

Module 1 (Calendar Module) contains formal interaction. Within this module, the participant (user) was supposed to make appointments with a companion. The user was instructed that Alexa could give information about the companion’s calendar on request. Under the first condition (without a companion), the user only got written information about the companion’s available dates and times. Under the second condition (with a companion), the companion, who had entered the room beforehand, could provide this information personally. As a result, the user was able to make appointments, negotiating with both Alexa and the companion. The latter human subject being part of the research team was properly instructed to interact only with the participant, not with Alexa.

Module 2 (Quiz Module) contains informal interaction. Within this module, the participant was supposed to answer quiz questions, e.g., “How old was Albert Einstein?”. The user was instructed that Alexa could not give a full answer. Instead, the SDS was able to offer informational support by solving small subtasks to find the answer. Under the first condition (without a companion), the participant was supposed to answer questions on his/her own. Under the second condition (with a companion), the participant, the companion, and Alexa built up a team. The companion (here again interacting only with the participant, not with Alexa) was instructed to make command proposals to the participant if the latter human subject got confused. In case of the companion presence, the quiz was more sophisticated compared to the solo scenario to encourage intensive cooperation between the team members.
3.2.2 Recording Setup

The recordings took place at the Institute for Information Technology and Communications, Cognitive Systems Group, Otto-von-Guericke-University Magdeburg. They were conducted in a living room-like surrounding so that the participants could get into a natural communication atmosphere that is difficult to achieve in a laboratory setting. A user was sitting on a sofa and interacting with the SDS placed on the table in front of him/her. A companion, who was present only in the two-person variants of each scenario, was sitting in the armchair at the same table. The interlocutors’ positions were fixed for all experiments to ensure full comparability. The interaction between the interlocutors is schematically illustrated by Figure 3.2.

As the SDS, the developers used an Amazon Alexa Echo Dot device of the second generation. They opted for a commercial system that was currently available on the market instead of a WOZ-simulated SDS in order to obtain a corpus that is more relevant to the capabilities of existing speech-based interfaces. For the same reason, Siegert, Krüger, et al. [2018] did not perform any video recording; most existing SDSs do not support video analysis. Furthermore, the participants’ awareness of being captured on video could have influenced their behaviour and their speaking styles in particular. Two high-quality neckband microphones (Sennheiser HSP 2-EW-3) and one shotgun microphone (Sennheiser ME 66) were used to record the user’s voice, the companion’s voice, and all the ambient sounds including the voice assistant’s output respectively. The recordings were stored in the WAV format with a sampling rate of 44.1 kHz and the 16-bit resolution.

3.2.3 Perceptual Experiments

Several psychological questionnaires accompanied the recording process. Before each recording session, a short self-defined questionnaire was conducted similarly to [Rösner et al., 2012] in order to collect socio-demographic information as well as the participants’ experience of interacting with
technical systems. After each recording session, some other self-defined computer-aided questionnaires were performed. The first two of them were focused on the participants’ impressions regarding a) their interaction with the voice assistant and the companion in general, b) possible changes in their speaking styles and prosody while interacting with the voice assistant or the companion. According to the so called principle of openness in examining subjective experience [Hoffmann-Rjem, 1980], questions were formulated starting from a higher openness level (non-restricted answering format), e.g., “If you compare your speaking style when interacting with Alexa or with the companion – can you recognise any differences? If yes, please describe these differences.” After that, the developers gradually changed the format of questions towards a lower openness level (structured answering format), e.g., “Did your speech tempo vary when interacting with Alexa or with the companion? Yes or No? If yes, please describe the differences.” Such a questionnaire scheme enables estimating to which degree the participants realised the vocal changes they introduced to their own speaking styles. If they specified detailed changes, e.g., melody and tempo, already in their non-restricted answers, a higher degree of awareness is indicated than if they reported on differences regarding these features only in their restricted answers to specific questions.

Another questionnaire was focused on the participants’ experience of using voice assistants. The AttrakDiff toolkit [Hassenzahl et al., 2003] was used to supplement the questions on self-evaluation with quantifying measurements of the H-M interaction quality in hedonistic and pragmatic aspects.

Answering all the questionnaires took about 20 minutes.

### 3.2.4 Corpus Characteristics

All the VACC participants were German speaking students of the Otto-von-Guericke-University Magdeburg. One experimental session with each of them took about 60 minutes (40 minutes for recording and 20 minutes for questionnaires). Table 3.6 summarises the corpus characteristics.

The participants came from different study courses, such as computer science, engineering science, and humanities. Therefore, the corpus is not biased towards technical students. All the participants had already known Amazon Alexa before, though only six of them specified that they had used Alexa on their own prior to this experiment: five participants used Alexa rarely and only for testing, and one participant specified that he used Alexa regularly – for playing music. Ten more participants confirmed their experience of using other voice assistants, such as Apple Siri, Google Assistant, and Microsoft Cortana: seven of them used these systems rarely, and only three participants used them regularly. In total, 18 out of 27 participants confirmed their experience of using voice assistants. The nine participants who had never used voice assistants before mistrusted the benefits of voice control due to privacy issues.

Siegert, Krüger, et al. [2018] also analysed the participants’ affinity to technology by asking how often each participant installed new software. There existed a clear division into two groups: 15 users with high affinity to technology, who familiarised themselves with new software at least once a quarter, and 12 users with low affinity to technology, familiarising themselves with new software only twice a year or less often. These groups turned out to be identical in terms of the joy of computer work, the ease of work by means of computers, the weekly duration of computer work, and the

---

2 The annotation was in progress as of the period of the experiments reported in the present study. The corpus version used within the present study contains a small part of conversations with 25 speakers out of 27 speakers. The latest corpus version is available from its developers (ingo.siegert@ovgu.de) for research purposes upon written request.
Table 3.6: General characteristics of VACC.

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDS type</td>
<td>Desktop SDS</td>
</tr>
<tr>
<td>Modalities</td>
<td>Audio; Text (manual transcripts)</td>
</tr>
<tr>
<td>Addressee labels</td>
<td>Machine-addressed; Human-addressed</td>
</tr>
<tr>
<td>SDS behaviour</td>
<td>real (non-acted)</td>
</tr>
<tr>
<td>Speakers’ behaviour</td>
<td>real (non-acted)</td>
</tr>
<tr>
<td>Speakers/sessions</td>
<td>27</td>
</tr>
<tr>
<td>Gender</td>
<td>Male: 13; Female: 14</td>
</tr>
<tr>
<td>Total recorded data</td>
<td>17 h 07 min</td>
</tr>
<tr>
<td>Annotated data</td>
<td>2 h 51 min</td>
</tr>
<tr>
<td>Experiment duration</td>
<td>Mean: 31 min</td>
</tr>
<tr>
<td>Age</td>
<td>Mean: 24; Std. dev.: 3.32; Min: 20; Max: 32</td>
</tr>
<tr>
<td>Language</td>
<td>German</td>
</tr>
<tr>
<td>Annotation</td>
<td>Utterance segmentation; Addressee; Transcription; Speaker events</td>
</tr>
<tr>
<td>Supplementary self-reports</td>
<td>Evaluation of interaction; AttrakDiff; Speaking style; Impressions from interacting with voice assistants</td>
</tr>
</tbody>
</table>

age of the first use of a computer. However, seven out of the nine participants having no experience with voice assistants also expressed less affinity to technology. This result is highly likely to be a matter of the participants’ curiosity. AttrakDiff is employed to understand how participants evaluate the usability and design of interactive products [Hassenzahl et al., 2003]. The following four usability aspects were specified: pragmatic quality (PQ), attractiveness (ATT), and hedonistic quality (HQ) including the subqualities identity (HQ-I) and stimulation (HQ-S). Regarding the users’ experience with Amazon Alexa, PQ, HQ-I, and ATT were perceived as neutral. Therefore, Alexa can be characterised as providing useful features, it allows the users to identify themselves with Alexa, and it seems attractive to them. Of course, all of these aspects may be improved. Regarding HQ-S, a slightly negative assessment can be observed, meaning that the support of the users’ individual needs was inappropriate or insufficient. This result can be explained by the high difficulty of the calendar task at which Alexa was error-prone. For all the four usability aspects, no significant differences between the experienced and the inexperienced users could be observed. Furthermore, the participants filled in questionnaires regarding their general impressions from the interaction with Alexa and the companion, regarding the changes in their speaking style and prosody during the interaction with both, and also regarding their previous experience of using voice assistants. Besides the structured part of these questionnaires, e.g., “Have you ever dealt with voice assistants besides Alexa? Yes or No?”, there were also unstructured ones which needed to be answered in arbitrary
form. According to the questionnaires, the participants were aware of differences in their speaking style, e.g., the length of sentences and the accentuation, when they were changing their addressees (Alexa or the companion).

VACC contains a great variety of paralinguistic phenomena, such as overlaps, laughter, engagement, and emotional reactions, and therefore can be used for other applications besides AD. More detailed information on VACC can be found in [Siegert, Krüger, et al., 2018].

3.3 Restaurant Booking Corpus

The Restaurant Booking Corpus (RBC) developed by Siegert et al. [2019] can be used for various purposes. However, the main idea behind its design was to obtain a data set of realistic H-H and H-M conversations with as few differences between these conversation types as possible. RBC thus represents the complexity-identical H-M AD problem (see Section 1.2.2 for more detail).

3.3.1 Corpus Design

A conversation scenario was chosen in such a way that it could be naturally performed by both a technical system and a human interlocutor using similar linguistic content. At the same time, the scenario supported a certain degree of flexibility to avoid word-for-word repetitions. Within this scenario, each participant (user) was planning several visits to various restaurants via a telephone-based booking system.

The participants were told that they would need to test different telephone-based SDSs and compare them with a human interlocutor in terms of interaction quality. The participants were also informed beforehand that they would talk to a human being or to one of two technical systems. The order of the interlocutors was permuted for each participant.

The conversation task was as follows: each participant was supposed to book tables at restaurants for dinner during a planned three-day trip with friends to an imaginary city called Quedlinstaett. The participant got a list of three recommended restaurants and completed one booking with each of the three interlocutors (one human being and two technical systems). Furthermore, the participant had to consider specific requirements: number of guests, desirable data and time, availability of vegan dishes, dishes of the day, possibility to sit outside, walking distance to the accommodation, and public transport availability. This task description was given to all participants in written form including the details of the trip.

A script sheet was designed to implement specific properties for each restaurant. This approach provided certain variations in phone calls made by the same participant. The differences were mostly confined to names, availability of free tables on specific dates, availability of vegan dishes, dishes of the day, possibility to sit outside, locations, and directions.

After the briefing, the participant was supposed to read the task description aloud. This was done on purpose to lower the first excitement at the beginning of the experiment and to record the reading participant’s voice. After that, the participant could freely choose the order of his/her calls. As soon as all three calls had been finished, the participant was supposed to answer two questionnaires. First, the AttrakDiff toolkit [Hassenzahl et al., 2003] was used to evaluate the participant’s interaction with
the three interlocutors and to maintain the cover story. Second, a short form of a self-defined questionaire similar to [Rösner et al., 2012] was applied to obtain the participant’s socio-demographic information as well as experience of using SDSs.

### 3.3.2 SDS Simulation

WOZ simulation was used to implement the two technical systems. For each restaurant, all possible system responses were preliminarily synthesised with MaryTTS (voice: bits3-hsm male German) based on the corresponding script. The first technical system (TS1) was assumed to understand single-slot requests only. The second technical system (TS2) was assumed to be more elaborate so that it could process multi-slot requests and back references. All system responses were played by a human operator simulating the SDS. Furthermore, the human interlocutor worked directly with the script to give answers similar to the technical systems. The telephone-like interface, which the participant used to make calls, was implemented as a simulated SIP-dialler.

### 3.3.3 Recording Setup

The recordings took place in a living room-like surrounding at the Institute for Information Technology and Communications, Cognitive Systems Group, Otto-von-Guericke-University Magdeburg. The aim of this setting was to allow the participants to get into a natural communication atmosphere that is difficult to achieve in a laboratory setting. Each participant was carrying a headset (Sennheiser Chat 3). In addition, a high quality shotgun microphone (Sennheiser ME 66) was employed to record the participant’s voice in high resolution. The same type of microphone was also used to record the human interlocutor’s voice. SDS responses as well as the output of the SIP-dialler were played to the user from the operator’s computer and recorded as well. All recordings were stored uncompressed as WAV-files with a sample rate of 44.1 kHz and the 16-bit resolution.

A scheme of the experimental setup is depicted in Figure 3.3.

![Fig. 3.3: Complexity-identical H-M AD problem represented in RBC.](image)

### 3.3.4 Corpus Characteristics

RBC contains recordings of 30 German speaking participants. All of them were students at the Otto-von-Guericke-University Magdeburg. They came from different study courses, such as computer science, engineering science, and humanities. Therefore, the corpus is not biased towards
technical students. One experimental session with one user took about 20 minutes (5 minutes for introduction, 10 minutes for recording, and 5 minutes for questionnaires). The manual annotation of each utterance comprises the interlocutor label (human or technical system), laughter, and other paralinguistic information. In addition, textual transcriptions were obtained using Google Cloud Speech API [Google LLC 2019] jointly with manual proofreading. Table 3.7 summarises the corpus characteristics. It can be additionally noted that phone calls involving a human interlocutor are significantly shorter than those with a technical system. Furthermore, the simple TS1 system has the longest call duration. However, the durations of the callers’ individual utterances are similar for the human interlocutor and TS2 and a bit shorter for TS1. These peculiarities can be explained by the experimental design: the human interlocutor was able to understand complex statements, and the more elaborate TS2 system could process at least combined requests in contrast to the simple TS1 system, which was able to perceive single-slot commands only. Therefore, the user had to split a single action into several commands to fill in all the slots while dealing with TS1. This fact led to shorter utterances and longer call duration for TS1.

Table 3.7: General characteristics of RBC.

<table>
<thead>
<tr>
<th>SDS type</th>
<th>Call centre SDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modalities</td>
<td>Audio; Text (manual transcripts)</td>
</tr>
<tr>
<td>Addressee labels</td>
<td>Machine-addressed; Human-addressed</td>
</tr>
<tr>
<td>SDS behaviour</td>
<td>WOZ</td>
</tr>
<tr>
<td>Speakers’ behaviour</td>
<td>real (non-acted)</td>
</tr>
<tr>
<td>Speakers/sessions</td>
<td>30</td>
</tr>
<tr>
<td>Gender</td>
<td>Male: 10; Female: 20</td>
</tr>
<tr>
<td>Total recorded data</td>
<td>5 h 37 min</td>
</tr>
<tr>
<td>Annotated data</td>
<td>1 h 40 min</td>
</tr>
<tr>
<td>Call duration</td>
<td>Mean: 193.6 ± 43.6 sec</td>
</tr>
<tr>
<td>Age</td>
<td>Mean: 23.97; Std. dev.: 3.45; Min: 18; Max: 31</td>
</tr>
<tr>
<td>Language</td>
<td>German</td>
</tr>
<tr>
<td>Annotation</td>
<td>Utterance segmentation; Addressee; Transcription</td>
</tr>
<tr>
<td>Supplementary self-reports</td>
<td>AttrakDiff; Human listeners’ addressee predictions</td>
</tr>
</tbody>
</table>

---

3 The annotation was in progress as of the period of the experiments reported in the present study. The corpus version used within the present study contains a small part of conversations with 30 speakers. The latest corpus version is available from its developers (ingo.siegert@ovgu.de) for research purposes upon written request.
3.3.5 Perceptual Experiments

AttrakDiff is usually employed to understand how participants evaluate the usability and design of interactive products [Hassenzahl et al., 2003]. This toolkit distinguishes three aspects: pragmatic quality (PQ), hedonistic quality (HQ), and attractiveness (ATT). The evaluation of these aspects has been performed for all three conversational partners. It can be noted that the human interlocutor is evaluated generally positively, the TS1 system is evaluated negatively, and the TS2 system is located somewhere between the other two services in terms of the perceived interaction quality. This ranking remains valid for almost all the characteristics being measured.

To evaluate humans’ ability of acoustic H-M AD, a perceptual experiment employing 10 native and 10 non-native German speaking listeners was conducted. Each listener was supposed to evaluate 450 utterances randomly selected from RBC and to solve a binary classification task, i.e., to classify each utterance as either human- or machine-directed. Utterances directed to TS1 or TS2 were merged into one class. The utterance selection included two stages. First, valid utterances were selected manually, skipping all samples with a direct hint at the interlocutor, samples containing a lot of laughter or other irrelevant paralinguistic phenomena. Utterances from the reading part of the corpus were skipped as well. Second, five utterances were randomly selected from the remaining samples for each speaker and for each interlocutor. The labelling was conducted using ikannotat2 [Siegert and Wendemuth, 2017]. Unweighted Average Recall (UAR) and Unweighted Average Precision (UAP) values obtained on this two-class problem are given in Table 3.8. They reveal that neither the native nor the non-native listeners could classify addressees properly. However, the native listeners demonstrated a considerably higher performance compared to the non-native ones, whose UAR and UAP were slightly above a chance level of 0.5. These results are expected given the experimental design.

3.3.6 Computational Experiments

The same two-class problem was solved using an automatic classifier. Utterances from the reading part and utterances containing a lot of laughter or other irrelevant paralinguistic phenomena were skipped as before. The Emobase feature set of the OpenSMILE toolkit was employed to extract acoustic features [Eyben et al., 2013]. Differences between the data samples of different speakers were eliminated using standardisation [Böck et al., 2017]. The extracted acoustic features were fed to an SVM with a linear kernel and a cost factor of 1. This classifier was implemented with WEKA [Witten et al., 2016]. A similar system had already been successfully used for AD and archived effective results ($UAR > 0.86$) [Siegert, Shuran, et al., 2018]. A Leave-One-Speaker-Out (LOSO) validation was applied and the overall UAR and UAP were calculated as the average over all speakers. The achieved results are given in Table 3.8. They show that machine-directed utterances remain hardly distinguishable from human-directed ones for the automatic classifier. The classification performance is just slightly above a chance level of 0.5. Again, this negative result is expected given the chosen experimental design.

More detailed information on RBC can be found in [Siegert et al., 2019].
Table 3.8: AD performance reached by human listeners and by an automatic classifier on RBC.

<table>
<thead>
<tr>
<th>Classification method</th>
<th>UAR</th>
<th>UAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native German speaking listeners</td>
<td>0.605 ± 0.033</td>
<td>0.601 ± 0.023</td>
</tr>
<tr>
<td>Non-native German speaking listeners</td>
<td>0.536 ± 0.048</td>
<td>0.534 ± 0.044</td>
</tr>
<tr>
<td>Automatic classifier</td>
<td>0.506 ± 0.002</td>
<td>0.506 ± 0.003</td>
</tr>
</tbody>
</table>

3.4 HomeBank Child/Adult Addressee Corpus

HomeBank Child/Adult Addressee Corpus (HB-CHAAC, further referred as HB for simplicity) represents the H-H AD problem (see Section 1.2.3 for more detail). HB was introduced within the Addressee Sub-Challenge of the 2017 ComParE Challenge [Schuller et al., 2017].

3.4.1 Corpus Design

The task of HB is to classify the following two categories of spoken utterances produced by adults: speech directed to a child and speech directed to another adult. Child-directed speech tends to have specific acoustic-phonetic and linguistic characteristics that distinguish it from adult-directed speech. Child-directed speech plays a critical role in facilitating language development and forming social behaviour [Soderstrom, 2007]. However, as of the HB release there have been few formal studies on discriminating these forms of speech computationally [Batliner, Schuller, et al., 2008; Schuster et al., 2014]. Furthermore, previous attempts to compare child-directed speech and adult-directed speech have been confined to highly constrained contexts. HB consists of conversations selected from a large repository of daylong real-world audio recordings known as HomeBank [Van-Dam et al., 2016]. A set of 20 conversations was selected from the following four corpora of the HomeBank repository: [Bergelson, 2016], [McDivitt and Soderstrom, 2016], [VanDam et al., 2016], and [Warlaumont and Pretzer, 2016].

3.4.2 Data Collection

All HomeBank recordings were conducted in the wild. In particular, the selected HB recordings were collected at 61 homes in total across four North American cities. These recordings feature North American English as the primary language being spoken, typically developing children, participants who granted their permissions to share the audio with the research community, and a distribution of children’ ages sampled as uniformly as possible between 2 and 24 months and across the four contributing corpora, with each child sampled only once. The recordings were collected using the LENA audio recorder, which was being worn by a child during each experimental session, and software [Greenwood et al., 2011] that provides automated identification of “conversational blocks”. Such blocks are audio chunks identified as speech bounded by 5 seconds of non-speech on either end. 1220 conversational blocks with a total duration of 42 hours 3 minutes were selected using this technique. A scheme of the experimental setup is depicted in Figure 3.4.
3.4.3 Data Annotation

Individual adult speakers’ audio clips within each conversational block (as identified by the LENA speaker diarisation algorithm) were then subjected to manual annotation for the challenge. Three trained human annotators classified each clip as child- or adult-directed. The annotators utilised both acoustic-prosodic information and context. Clips deemed to be non-speech, not produced by adults, or ambiguous between the two addressee categories were excluded from consideration. All the remaining clips were additionally labelled with the speaker’s gender label by a human annotator. The annotators achieved high reliability (in terms of Fleiss’ kappa) at classifying addressees ($\kappa > 0.75, p < 0.001$).

The HB data splitting (for the challenge) and the general characteristics of the corpus are specified in Table 3.9 and 3.10 respectively. The challenge baseline is described in [Schuller et al., 2017]. More detailed information on HB can be found in [Schuller et al., 2017; Casillas et al., 2017].

Table 3.9: HB partitions of the Interspeech 2017 ComParE Challenge.

<table>
<thead>
<tr>
<th>Addressee</th>
<th>Partitions</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Development</td>
</tr>
<tr>
<td>Child</td>
<td>2302</td>
<td>2182</td>
</tr>
<tr>
<td>Adult</td>
<td>1440</td>
<td>1368</td>
</tr>
<tr>
<td>Sum</td>
<td>3742</td>
<td>3550</td>
</tr>
</tbody>
</table>

4 See https://osf.io/d9ac4/ for more detail.
5 The metadata concern the corpus version introduced at the Interspeech 2017 ComParE Challenge. This corpus version is utilised in the present study. The labels of the test set remain unavailable as no one of the challenge participants has managed to surpass the challenge baseline. The latest corpus version can be obtained from its developers (https://homebank.talkbank.org/) upon request.
Table 3.10: General characteristics of HB.

<table>
<thead>
<tr>
<th>SDS type</th>
<th>Passive listener</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modalities</td>
<td>Audio</td>
</tr>
<tr>
<td>Addressee labels</td>
<td>Child-addressed; Adult-addressed</td>
</tr>
<tr>
<td>Speakers’ behaviour</td>
<td>real (non-acted)</td>
</tr>
<tr>
<td>Speakers</td>
<td>unknown</td>
</tr>
<tr>
<td>Total recorded data</td>
<td>42 h 3 min</td>
</tr>
<tr>
<td>Annotated data available for the challenge</td>
<td>3 h 13 min</td>
</tr>
<tr>
<td>Language</td>
<td>North American English</td>
</tr>
<tr>
<td>Annotation</td>
<td>Segmentation; Addressee; Gender</td>
</tr>
</tbody>
</table>

3.5 Tools and Computational Infrastructure

Within our study, we use some off-the-shelf solutions to facilitate the development process. They are briefly described in the present section below.

3.5.1 Acoustic Feature Extraction

We apply the IS13_ComParE feature configuration of the OpenSMILE toolkit to extract acoustic-prosodic information from audio. The IS13_ComParE functionals serve as input for some of our models, while the others employ the IS13_ComParE LLDs. The IS13_ComParE LLDs and their functionals are briefly described in Section 2.2.1. See [Eyben et al., 2010; Eyben et al., 2013] and [Eyben, 2015] for more detail on the OpenSMILE toolkit and its feature configuration respectively. Voice activity detection is conducted using the webrtcvad package for Python. We also employ the WebMAUS tool [Kisler et al., 2017] to align speech with text if necessary.

3.5.2 ASR Feature Extraction

The first step of text analysis is speech recognition. If manual transcripts are given, we deal with them. Otherwise, the Google Cloud Speech API is applied to recognise users’ speech [Google LLC 2019]. After that, we perform text preprocessing (stemming and stop-word filtering) with R [Feinerer, 2013] and extract numeric information from the obtained text. The T2V representations, explained in Section 2.2.3, are obtained with our own feature extractors written in C# and based on [Sergienko, 2016]. The W2V representations, described in the same section, are extracted applying the Spacy toolkit [Spacy 2019]. The same toolkit is also used for POS tagging and dependency parsing.
### 3.5.3 ASR Metafeature Extraction

Alongside with the ASR textual output, we leverage some additional information, e.g., the number of different ASR hypotheses and the probability of the top hypothesis, from the acoustic and the language model of Google Cloud Speech API [Google LLC 2019]. See Section 2.2.4 for more detail on various ASR metafeatures.

### 3.5.4 Visual Feature Extraction

We extract visual features and perform face detection with the OpenCV library for Python [OpenCV 2019].

### 3.5.5 Feature Selection and Feature Transformation

PCA and RFE, described in Section 2.3.1 and 2.3.2 respectively, are implemented using the RapidMiner software [Hofmann and Klinkenberg, 2016].

### 3.5.6 Classification

All ANNs developed within the present study are implemented using the Keras library [Chollet et al., 2019] with the TensorFlow backend [Abadi et al., 2016] for Python. See Section 2.4.4 for more detail on various ANN architectures. The Rocchio classifier, described in Section 2.4.3, is implemented in C++. Other classifiers are developed using RapidMiner. The own software is implemented with C++ [Oualline, 2003], C# [Hejlsberg et al., 2010], or Python [Lutz, 2013]. System scripts are written in Bash [Cooper, 2014] or Batch [Microsoft Corporation 2019].

### 3.5.7 Computational Infrastructure

All computations involving ANNs are conducted on a Graphics Processing Unit (GPU, GeForce GTX 1050) of a personal computer running the 64-bit version of the Windows 10 operational system. Other simulations are performed on Central Processing Units (CPUs) belonging to the computational cluster of the Institute of Communications Engineering, Ulm University, Germany.

### 3.6 Summary

We have chosen the four aforementioned corpora for the following reasons. First, they contain realistic, spontaneous conversations between several humans and a special interlocutor, i.e., all of the corpora are appropriate for AD. Second, each of the corpora contains at least audio data. Therefore, we can work with at least two modalities: speech and text. Furthermore, SVC also contains video data. Third, SVC, VACC, and RBC were recorded in German, while HB was uttered in English. This fact allows us to investigate AD for the German language and also perform a cross-language study for German and English. Fourth, the corpora model different AD problem statements in various scenarios that allows us to check our classifiers under different conditions. SVC and VACC are dedicated to the classical H-M AD problem with a portable SDS and with a desktop SDS respectively.
RBC models the complexity-identical H-M AD problem with a call centre SDS. HB represents the H-H AD problem with a passively listening SDS. The descriptions of the AD problem statements can be found in Section 1.2. The most important peculiarities of the considered corpora are summarised in Table 3.5, 3.6, 3.7, and 3.10 for SVC, VACC, RBC, and HB respectively.

All the four corpora are used for cross-corpus experiments on acoustic and textual AD in Chapter 4. Chapter 5 is dedicated to the relationship between the individual problem statements. Section 5.4 is focused on the relationship between the classical H-M AD (VACC and SVC) and the H-H AD problem (HB). The relationship between the classical (VACC and SVC) and the complexity-identical H-M AD problem (RBC) is analysed in Section 5.5. SVC is thoroughly investigated using approaches to acoustic, textual, and visual H-M AD in Chapter 6.
Modern SDSs face AD problems in various scenarios and domains. Therefore, it is important to train and test an AD system on several corpora having different characteristics to check its AD performance under various conditions. In the present chapter, we conduct a cross-corpus study on acoustic and textual AD using the four corpora (VACC, SVC, RBC, and HB) described in Chapter 3. They represent various SDSs with different capabilities in different domains, talking to which humans also behave differently. We determine various AD aspects which are common for different corpora, e.g., the optimal temporal context length for audio and text, and also those aspects which are corpus-specific, e.g., relevant acoustic features.

Let us point out the main similarities and differences between the corpora. First, each of the four corpora contains at least audio data. Second, all the corpora except HB were recorded in German, while the latter corpus was uttered in English. Third, they all represent binary classification problems. Despite the different domains, VACC and SVC are quite similar in terms of their AD scenarios. Furthermore, they both represent the classical H-M AD problem (see Section 1.2.1, 3.1, and 3.2 for more detail). However, RBC was recorded for a slightly different classification problem of complexity-identical H-M AD (see Section 1.2.2 and 3.3 for more detail), while HB was created for a completely different problem of H-H AD (see Section 1.2.3 and 3.4 for more detail). In this light, we expect our classifiers to generalise addressee patterns across VACC and SVC. However, it is an open question whether RBC and HB complement the other two corpora. This question is answered in Chapter 5. SVC is used for some experiments in the present chapter and Chapter 5, while Chapter 6 is exclusively dedicated to experiments with this corpus.

We assume the four corpora to fully complement one another in terms of their data examples and class labels. In other words, we can merge all four corpora and classify their examples as either directed to some special addressee, such as a machine or a child, or to someone else. This assumption can be straightforwardly applied to VACC, SVC, and RBC since they have the same utterance labels: human-directed and machine-directed. For HB, we suppose that adult-directed utterances are equivalent to human-directed ones and child-directed utterances are similar to machine-directed ones. The idea behind this assumption is explained in Section 1.2.3.

The present chapter is organised as follows. In Section 4.1, we propose several classifiers for acoustic and textual AD. In Section 4.2, we discuss the metrics that are employed to measure the classification performance of the proposed models. In Section 4.3, we split the corpora into training, development, and test data partitions that are used in our study. Experiments on acoustic and textual
AD are reported in Section 4.4 and 4.5 respectively. The content of the present chapter is summarised in Section 4.6.

4.1 Proposed Classifiers

People tend to change their manner of speech as soon as they start talking to some special addressee, such as an SDS [Batliner, Hacker, and Nöth, 2008] or a child [Batliner, Schuller, et al., 2008], since they do not perceive this interlocutor as an adequate conversational partner. As a result, humans accommodate certain aspects of their speech to such an interlocutor, making the speech more rhythmic, louder, and generally easier to understand as if they were talking to a person who is hard of hearing or having communicational difficulties. People may also use a specific lexicon or syntax when they address a special interlocutor. Another motivation for adult speakers addressing children is to attract the interlocutors’ attention that is usually not easy. In this light, we tackle AD as a speech and text classification problem. Within the present chapter, we propose the following four models for AD: ComParE_func, ComParE_LLD, ASR_meta, and Sem_BLSTM. These classifiers are described in the present section below. ComParE_func and ComParE_LLD employ acoustic-prosodic features which do not require any ASR and may therefore facilitate portability across domains and even languages [Shriberg et al., 2013]. ASR_meta and Sem_BLSTM use ASR metafeatures and transcripts respectively. Designing the architectures of the proposed models, we search for a trade-off between the model complexity and the input feature complexity. In other words, the more complex, abstract features we extract from our data, the simpler model is sufficient to process them, and vice versa; the lower-level features are employed, the more advanced model is required to deal with them.

There exist speech signal representations designed specially for acoustic H-M AD [Shriberg et al., 2013]. The machine-directed utterances from the corpus used by Shriberg et al. [2013] were short, simple phrases. However, the machine-directed utterances from the data that we have at our disposal were recorded under real-life conditions and usually contain whole sentences of spontaneous speech. Furthermore, it is an open question how the specific attributes proposed by Shriberg et al. [2013] perform on H-H AD and complexity-identical H-M AD. Therefore, we would not like to confine to such a narrow attribute set. Instead, we want to use the IS13_ComParE features in order to capture all the variety of spontaneous speech.

The IS13_ComParE LLDs and their functionals were shown to be a universal solution for a wide range of paralinguistic problems besides acoustic AD, e.g. acoustic emotion recognition [Fedotov et al., 2018], native speech detection, and neurological pathology estimation [Schuller et al., 2015]. The IS13_ComParE functionals in combination with a linear SVM were applied to the problem of acoustic H-H AD on HB by Schuller et al. [2017].

4.1.1 ComParE_func Classifier

The first classifier (ComParE_func) is based on a simple SVM with a linear kernel (Formula 2.100) [Cortes and Vapnik, 1995]. This model depicted in Figure 4.1 receives a large set of utterance-level paralinguistic features as input and returns an addressee prediction for the whole utterance. In our case, more sophisticated kernels (Formula 2.101 – 2.106) do not provide any considerable classification improvement while increasing the computational time noticeably. The simplicity of
4.1 Proposed Classifiers

the linear kernel (Formula 2.100) is compensated by the quantity and complexity of the acoustic features being used. On the one hand, the higher dimensionality a feature space has, the higher the chance is that our data points turn out to be linearly separable in this space. On the other hand, the dimensionality curse should also be taken into account.

As the input attribute set, we employ the IS13_ComParE configuration of 6373 functionals computed over 130 LLD contours [Eyben, 2015]. These LLDs reflect various energy-related, spectrum-related, and voicing-related acoustic characteristics described in Section 2.2.1. In our study, the IS13_ComParE functionals are extracted using the OpenSMILE toolkit [Eyben et al., 2010] at the utterance level. Prior to classification, the feature vectors are normalised by computing the standard score on the training set of each corpus: each feature is brought to zero mean and unit variance. Additionally, we fine-tune the SVM complexity parameter $C$ on a development set.

Though speaker normalisation could potentially improve the classification results [Batliner, Hacker, and Nöth, 2008], using this approach automatically means that our system implicitly "knows" all speech produced by unknown speakers beforehand. Furthermore, there are no individual speakers specified for HB. In order to make our results fairer, we apply corpus normalisation instead of speaker normalisation. Alternatively, we could train a speaker identification model on the speakers from the training set. This model would be able to identify the speaker’s profile "nearest" (in terms of confidence scores obtained for the speakers from the training set) to the actual speaker of the current test utterance and to normalise the feature vector of this utterance in accordance with this speaker’s profile. However, speaker identification is another challenging problem that lies beyond the scope of the present thesis.
4.1.2 ComParE_LLD Classifier

The second classifier (ComParE_LLD) depicted in Figure 4.2 is a more advanced version of ComParE_func. It is an RNN that consists of two stacked LSTM layers followed by a global max pooling, a dropout [Srivastava et al., 2014], and a dense (fully connected) layer. The latter layer contains two neurons (one for each class) and has the softmax activation function that is applied after a batch normalisation operator. Using simple RNNs instead of the LSTMs worsened the classification performance. See [S. Hochreiter and Schmidhuber, 1997] and Section 2.4.4 for more detail on LSTMs.

As input, the neural network receives the same LLD sequences used for computing the IS13_ComParE functionals for the ComParE_func model. In contrast to the functionals, the LLD sequences are time-continuous by nature. In other words, we let the ComParE_LLD classifier learn its own "functionals" from the time-continuous input. Each sequence element is a vector of 130 LLDs extracted for a sliding time window of 60 milliseconds with an overlap of 50 milliseconds. Prior to classification, the feature vectors are normalised by computing the standard score on the training set of each corpus: each LLD is brought to zero mean and unit variance. The features are extracted from sliding time windows of various lengths (from 1/8 to 8 seconds). The context windows are cut out of audio files with an overlap of 75%. The windows containing no speech are excluded from consideration. In other words, the ComParE_LLD classifier learns to predict addressees at the window level. The segmentation into acoustic context windows is performed for the following two reasons. First, it allows us to increase the number of training examples. Second, it enables experiments with various context window lengths. These experiments can shed light on the optimal context length that is sufficient for reliable acoustic AD (see Section 4.4.3 for more detail). The predictions obtained on several context windows belonging to one utterance are averaged to get the final utterance-level prediction.

As the result of a trial-and-error procedure, we have chosen the following hyperparameters for the ComParE_LLD classifier: two layers of 128 LSTM units each, Gaussian noise applied to the input signal if mixup is deactivated, a dropout of 20% if mixup is deactivated, cross-entropy as a loss function, Adam [Kingma and Ba, 2014] as a weight optimisation algorithm, 100 epochs, and a mini-batch size of 32 examples. The initial learning rate is selected from the set {10^{−3}, 10^{−4}, 10^{−5}} and then divided by 10 if there is no performance improvement observed for the past 10 epochs on the development set. We make checkpoints, saving the current model weights at each epoch and using the best checkpoint as the resulting model according to its performance on the development set.

4.1.3 ASR_meta Classifier

The architecture of the third classifier (ASR_meta) depicted in Figure 4.3 is similar to ComParE_func. Both models are based on SVMs dealing with utterance-level features. As input, the ASR_meta classifier receives a vector of utterance-level metafeatures obtained from ASR acoustic and language models [Google LLC 2019]. This vector consists of the following four components:

- Number of hypotheses;
- Confidence of the top hypothesis;
- Number of words in the top hypothesis;
- Utterance duration.
These features represent the degree to which the ASR is confident of its recognition hypothesis. People try speaking more clearly than usual when addressing an SDS, and therefore machine-directed speech tends to match the ASR acoustic and language patterns better compared to human-directed speech, resulting in a higher ASR confidence [Shriberg et al., 2012]. Prior to classification, the input features are normalised by computing the standard score on the training set of each corpus: each
feature (except the confidence of the top hypothesis as it is already normalised by nature) is brought to zero mean and unit variance. ASR metainformation can also be useful for improving the overall performance of an SDS [Lopes et al., 2012].

As the result of a trial-and-error procedure, we have chosen a radial kernel (Formula 2.102) for the ASR_meta classifier instead of the linear one (Formula 2.100) that is employed in Com-ParE_func. Additionally, we fine-tune the SVM complexity parameter $C$ on a development set.

### 4.1.4 Semantic Classifier

We employ word-level text representations together with a recurrent model (Sem) to classify users’ utterances. This model is depicted in Figure 4.4. There has already been an attempt to apply a recurrent model to textual H-M AD on a simpler corpus [S. Ravuri and Stolcke, 2015]. Although SVC originally includes word-level addressee labels, we transform them into utterance-level ones; we define an utterance-level label as the most frequent word-level label in the utterance as specified by Batliner, Hacker, and Nöth [2008]. This transformation is conducted to ensure compatibility with the other corpora in our cross-corpus experiments.

For building word representations, we use the GloVe algorithm proposed by Pennington et al. [2014] (see Formula 2.62 and 2.63 for more detail). It is an unsupervised algorithm for learning global word representations based on word co-occurrences in a large amount of textual data, hence the built word embeddings are domain-independent and can be used in a wide range of applications.
The performance reached with GloVe word representations is very similar to the performance of Word2Vec embeddings in most applications. However, the process of training the GloVe representations can easier be parallelised since it is essentially based on matrix factorisation.

Besides the three aforementioned German corpora (VACC, SVC, and RBC), we utilise the following two large out-of-domain textual corpora to train GloVe word embeddings for German: the TIGER [Brants et al., 2004] and the WikiNER Corpus [Nothman et al., 2013]. We have chosen the resulting dimensionality of the word embeddings to be equal to 300 in accordance with an existing model for German built in the spaCy toolkit being used for learning the word representations [Spacy 2019].

In order to perform text classification on the English corpus (HB), we use the pretrained English word embeddings contained in the spaCy toolkit.

After extracting the word embeddings, each utterance is represented as a sequence of word vectors. This sequence is processed with a recurrent model. S. Ravuri and Stolcke [2015] compare applications of a simple RNN and an LSTM to textual H-M AD. The authors show that simple RNNs perform better on one data set containing short utterances (the median of the utterance length distribution is below 5 words), while LSTMs demonstrate a higher classification performance on another data set that consists of long utterances (the median of the utterance length distribution is above 10 words). Utterance length distributions of the considered corpora are depicted in Figure 4.5 – 4.8 and their properties are summarised in Table 4.1. According to S. Ravuri and Stolcke [2015], we should have employed LSTMs for SVC (as its utterance length distribution has a median of 9 words) and simple RNNs for the other corpora (since their utterance length distributions have median values of around 5 words each). However, the utterance lengths significantly vary over each corpus, and therefore we have to pad each utterance with zero vectors at the end to the length of the longest utterance in our data collection. There exists a masking layer in Keras that handles padded empty time steps properly. However, the GPU-optimised layers in the existing versions of Keras and TensorFlow compatible with CUDA 8.0 (the latest driver version supported by our GPU hardware) do not support masking. This fact results in padded empty time steps which are not ignored by the model and hence influence the model output. Using conventional layers instead of those designed for GPU is undesirable as the conventional layers are significantly slower than their GPU-optimised analogues. As a result, we obtain utterances of the same length equal to several dozens of words. In fact, many of these words are zero vectors that do not contain any valuable information. It turned out that LSTMs coped with such a padded input better than simple RNNs did. Taking these results into account, we apply a BLSTM network (a bidirectional version of LSTM) to all of our AD problems. A conventional LSTM can also be used, though the unidirectional LSTM showed a lower classification performance compared to the BLSTM in our preliminary experiments. See [S. Hochreiter and Schmidhuber, 1997] and Section 2.4.4 for more detail on LSTMs.
Fig. 4.4: Extended version of the Sem_BLSTM classifier. This version is applied in Chapter 6. The model version used in Chapter 4 and 5 differs from the depicted model in the following way. First, the mask is disabled. Second, the BLSTM layer returns only the last member of the output sequence. Third, the RNN component returns a single utterance-level prediction. Fourth, the postprocessing component is disabled.
**Fig. 4.5:** Kernel density estimation of the VACC utterance length distribution for text.

**Fig. 4.6:** Kernel density estimation of the SVC utterance length distribution for text.
Fig. 4.7: Kernel density estimation of the RBC utterance length distribution for text.

Fig. 4.8: Kernel density estimation of the HB utterance length distribution for text.
Table 4.1: Properties of the utterance length distributions visualised in Figure 4.5 – 4.8. All values are given in words.

<table>
<thead>
<tr>
<th>Corpus name</th>
<th>Addressee label</th>
<th>Mode</th>
<th>Median</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VACC</td>
<td>Machine</td>
<td>1.000</td>
<td>4.000</td>
<td>4.160</td>
<td>3.364</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>2.000</td>
<td>6.000</td>
<td>6.641</td>
<td>5.646</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.000</td>
<td>4.000</td>
<td>4.854</td>
<td>4.279</td>
</tr>
<tr>
<td>SVC</td>
<td>Machine</td>
<td>9.000</td>
<td>9.000</td>
<td>9.466</td>
<td>5.137</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>6.000</td>
<td>9.000</td>
<td>10.351</td>
<td>7.685</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>9.000</td>
<td>9.000</td>
<td>9.912</td>
<td>6.561</td>
</tr>
<tr>
<td>RBC</td>
<td>Machine</td>
<td>2.000</td>
<td>5.000</td>
<td>6.183</td>
<td>4.918</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>3.000</td>
<td>5.000</td>
<td>7.238</td>
<td>6.306</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>2.000</td>
<td>5.000</td>
<td>6.530</td>
<td>5.437</td>
</tr>
<tr>
<td>HB</td>
<td>Child</td>
<td>2.000</td>
<td>4.000</td>
<td>4.452</td>
<td>3.306</td>
</tr>
<tr>
<td></td>
<td>Adult</td>
<td>3.000</td>
<td>5.000</td>
<td>5.991</td>
<td>5.420</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>3.000</td>
<td>4.000</td>
<td>5.054</td>
<td>4.326</td>
</tr>
</tbody>
</table>

The BLSTM-based text classifier (employed in Chapter 4 – 6) and its LSTM-based analogue (used in Chapter 6 only) are denoted as Sem_BLSTM and Sem_LSTM respectively. For each input sequence, a single recurrent layer of the model returns only the last member of the output sequence. The recurrent layer is followed by a dropout [Srivastava et al., 2014] and a dense (fully connected) layer. The latter layer contains two neurons (one for each class) and has the softmax activation function that is applied after a batch normalisation operator. As a result, the network receives a textual utterance as input and returns an addressee prediction as output. In other words, the Sem classifier learns to predict addressees at the utterance level. The entire architecture of the Sem_BLSTM classifier is visualised in Figure 4.4.

We use the following network hyperparameters optimised on a development set: one recurrent layer of 30 LSTM or 15 BLSTM units (both versions have approximately the same number of learnable parameters), cross-entropy as a loss function, RMSprop as a weight optimisation algorithm [Tieleman and Hinton, 2012], a learning rate of 0.01, 20 epochs, 20% dropout, and a mini-batch size of 32. We also make checkpoints, saving the current model weights at each epoch and using the best checkpoint as the resulting model according to its performance on the development set.

4.2 Metrics

We use various loss functions for training our models. However, it would be reasonable to represent our results in terms of a single metric to ensure their comparability with one another. Furthermore,
this metric should be chosen in such a way to ensure the comparability with the existing baselines for SVC [Batliner, Hacker, and Nöth, 2008], RBC [Siegert et al., 2019], and HB [Schuller et al., 2017]. Taking this into account, we have chosen Unweighted Average Recall (UAR) as the classification performance metric.

In ML, the recall of a class is the ratio of the number of the correctly classified data points belonging to the class to the total number of the data points in this class. The formal definition of recall is as follows:

\[
R_c = \frac{|A_c \cap B_c|}{|A_c|},
\]

where \(C\) is a set of all classes, \(c\) is the class for which we calculate the recall value \(R_c\), \(A_c\) is the set of all data points which actually belong to the class \(c\), and \(B_c\) is the set of the data points classified as belonging to the class \(c\). There exist several approaches to averaging the recall values of individual classes. Unweighted average recall means a simple averaging that can be formally described as follows:

\[
R_{UAR} = \frac{\sum_{c \in C} R_c}{|C|}.
\]

The minimal value of \(R_{UAR}\) is hence equal to 0.5 for a two-class problem. Such a low performance means that either the classifier makes random predictions or it suffers from overfitting and assigns all examples to one class.

### 4.3 Data Splitting

In the VACC, RBC, and HB data, each utterance has an addressee label. In the SVC data, each word has its own addressee label. However, some of our models require the data to have utterance-level labels for classification. Given that SVC, like the other corpora, has already been segmented into utterances, we define an utterance-level label as the most frequent word-level label in the utterance as specified by Batliner, Hacker, and Nöth [2008]. Other models, e.g., ComParE_LLD, deal with acoustic context windows extracted from the original audio utterances. Such a window is processed as an individual example and therefore needs its own addressee label as well. The labelling of acoustic context windows is not a problem for VACC, RBC, and HB; all windows belonging to the same utterance also have the same addressee label. In case of SVC, we apply the WebMAUS tool to align speech with text [Kisler et al., 2017]. This alignment allows us to extract window-level labels based on the word-level labels.

We split each corpus into a training, a development, and a test set in order to apply a universal experimental scheme to all our models. The training (58 speakers) and the test partition (remaining speakers) of SVC have already been defined by the corpus developers [Batliner, Hacker, and Nöth, 2008]. We tried to reproduce them to ensure the comparability of our results with existing baselines. However, the authors specified no development partition for SVC, and therefore we use about 20% of the speakers from the initial training set as a development set. Likewise Batliner, Hacker, and Nöth
[2008], we merge ROT, POT, and SOT into one class and classify it versus NOT (see Section 3.1.2 for more detail on the class labels).

The partitions of HB were also predefined by the organisers of the Interspeech 2017 ComParE Challenge [Schuller et al., 2017]. We keep these partitions to ensure the comparability of our results with existing baselines. Unfortunately, the HB test set is useless for us since the challenge, including the Addressee Sub-Challenge, has already been finished. Given that none of the participants managed to surpass the Addressee Sub-Challenge baseline, the challenge organisers have not revealed the labels of the HB test set for public research yet. Therefore, we use the initial HB development set as a new test set and utilise about 20% of the utterances from the initial training set as a new development set.

Given that the partitions of SVC and HB are predefined, we split the other two corpora (VACC and RBC) in a similar way, trying to keep the proportion of speakers (or utterances in case of HB, in which individual speakers are not specified) equal to 5/1/4 for the train, the development, and the test set respectively. In case of RBC, we merge TS1-directed utterances and TS2-directed ones into one class and classify it versus human-directed utterances likewise Siegert et al. [2019] (see Section 3.3.2 for more detail on the class labels). The partitions of the four considered corpora are presented in Table 4.2. It is difficult to maintain the proportion of the target classes equal for all the data sets. The target classes are well balanced resulting in a nearly ideal proportion of 1/1 for SVC. However, machine/child-addressed utterances predominate in the other corpora; we would have had to neglect a considerable part of their data examples in our attempt to make the data well balanced that is undesirable given the relatively small amount of the data at our disposal. We show in our experiments that the imbalanced data does not appear to be a critical problem for AD. Furthermore, in Chapter 5 we introduce a data augmentation method based on the mixup approach proposed by H. Zhang et al. [2017]. Due to a soft targeting technique, our method resolves the imbalanced data issue. The method is also demonstrated to improve AD on VACC and RBC in Chapter 5.

4.4 Speech Analysis

Prior to speech analysis, all audio files of the four corpora were brought to a sampling rate of 16 kHz for compatibility reasons, although some of the corpora had audio files of higher sampling rates. Downsampling to 8 kHz accelerated the computations but did not result in any noticeable changes in terms of classification performance.

4.4.1 Keep-One-Corpus-In Experiments

We conduct a series of cross-corpus experiments with the ComParE_func classifier, applying a Keep-One-Corpus-In (KOCI) scheme. This scheme is the antipode of the well-known Leave-One-Corpus-Out (LOCO) scheme. Within the latter scheme, the model is trained on a mixture of all corpora except one and then tested on each of the corpora including that one excluded before. According to the KOCI scheme, the model is trained on one corpus and then tested on each of the corpora including the training corpus.

Within the present section, the ComParE_func classifier is trained and tested on the data partitions from Table 4.2. Results of KOCI experiments on all four corpora are visualised in Figure 4.9.
Let us represent the UAR values from this figure as a matrix A with elements $a_{\text{train_corpus, test_corpus}}$. According to the design of the KOCI experiments, the diagonal elements should be greater than a chance performance of 0.5 and also considerably greater than the off-diagonal elements. Large values of some off-diagonal elements indicate relationships between the corresponding corpora; the higher the off-diagonal value is, the stronger is the relationship.

First of all, it can be noted that $a_{\text{VACC}, \text{VACC}}$ and $a_{\text{SVC}, \text{SVC}}$ are significantly greater than the other elements of A. In other words, the acoustic AD problems of VACC and SVC turned out to be essentially easier for our ComParE_func classifier than those of the other two corpora. In addition, $a_{\text{VACC}, \text{SVC}}$ and $a_{\text{SVC}, \text{VACC}}$ are significantly greater than the other off-diagonal elements of A. The latter means a clear relationship between VACC and SVC in the acoustic-prosodic sense. This result motivates us to explore the potential of the cross-corpus data augmentation on these two corpora by means of mixup and deep learning in Section 5.4.

Although RBC is the most challenging corpus for our classifier, this corpus also demonstrates some relationship with VACC: $a_{\text{RBC}, \text{VACC}}$ is noticeably above 0.5 and also above $a_{\text{RBC}, \text{RBC}}$. At the same time, the symmetrical element $a_{\text{VACC}, \text{RBC}}$ is just slightly above the chance level. In other words, the model trained on RBC performs better on VACC than on the RBC test set. At first sight, it seems to be a contradiction. To resolve it, we need to recall the RBC experimental setup described in Section 3.3. The acoustic addressee patterns captured within RBC are extremely subtle since visual and contextual differences between a human addressee and a machine were minimised for the user. This gives us the following explanation for the contradiction arisen: the training on the complexity-
Fig. 4.9: KOCI experiments with acoustic AD on all four corpora. All results are presented in terms of UAR.

identical H-M AD setup of RBC allows the model to learn true acoustic addressee patterns that are also typical for the classical H-M AD setup of VACC. However, the opposite statement does not work due to collateral factors, such as differences between the human addressee’s appearance and the SDS appearance, between the domains and complexity levels of human-directed and machine-directed utterances, etc. These differences influence the user’s behaviour, particularly speech and prosody, within the VACC setup, but not within the RBC setup, and cause the model to overfit. SVC and RBC do not exhibit such a relationship probably due to the difficult acoustic conditions of SVC; in contrast to VACC, which consists of indoor recordings, SVC additionally contains outdoor recordings made with a distant microphone. RBC and its relationships with VACC and SVC are thoroughly investigated in Section 5.5.

The matrix A does not reveal any relationship between HB and the other three corpora. This corpus is thoroughly investigated in Section 5.4.

4.4.2 Acoustic Feature Selection Experiments

In the present section, we conduct another series of experiments with the ComParE_func classifier, aiming to estimate the degree of similarity between our corpora in the sense of individual acoustic features. After the functional extraction with the IS13_ComParE configuration, we perform RFE, using the coefficients of the normal vector of a linear SVM as attribute weights. The core idea of this approach is shown in Figure 4.10. The maximum margin hyperplane is depicted as a red line. The minimal component $w_1$ of the normal vector $w$ of the maximum margin hyperplane explains the fewest differences between the target classes, and therefore the corresponding dimension $x_1$ of the current feature space can be excluded out of consideration. The process is repeated for lower-dimensional feature spaces until an effective trade-off between the feature space dimensionality and
the desired classification performance is reached. The problem of RFE becomes trivial in a two-
dimensional feature space. RFE results in increasing information losses for low-dimensional feature
spaces.

We have chosen the RFE in combination with a linear SVM for the following reasons. The
first one is its low computational complexity. The second reason is its appropriateness for large
feature sets; the data set being analysed with PCA should usually contain at least ten times as many
examples as the feature set dimensionality is so that the method can show adequate results in real
applications. Third, RFE is a feature selection method in contrast to PCA, which performs feature
transformation. Therefore, the results obtained with RFE are easier to interpret and also to compare
with one another on different data sets than those obtained with PCA. The experiments described in
the present section are aimed at investigating similarities between different corpora in terms of their
relevant acoustic features and not at reaching the highest possible AD performance. RFE is ideal for
this purpose.

Figure 4.11 demonstrates RFE curves obtained by applying ten-fold Leave-One-Speaker-Group-
Out (LOSGO) cross-validation on each corpus. The resulting classification performance is calcu-
lated as UAR for comparability with the existing studies and averaged over all folds for each re-
duced feature set. A feature set is considered to be optimal if further RFE iterations lead to stable
performance losses. For each corpus, we choose one optimal feature set obtained on a random fold
and analyse their intersection depicted in Figure 4.12.

It can be noted in Figure 4.11 that each curve flattens beyond a certain dimensionality threshold,
meaning that the method has eliminated a large number of irrelevant attributes and correlated fea-
tures. The representative acoustic functionals vary significantly: VACC, SVC, RBC, and HB have
only 1170, 2020, 1570, and 400 relevant features out of 6373 respectively, while having only 17
attributes in common:
4.4 Speech Analysis


These spectrum-related features are used to identify the notes, pitch, rhythm, and melody. The IS13_ComParE feature notations can be found in [Eyben, 2015] and Section 2.2.1. Curiously enough, the common feature set does not contain any energy-related features, meaning that loudness is not that important for acoustic AD or it varies over different corpora significantly. If taken alone, the selected features do not provide reliable AD results on our corpora. Apparently, the HB corpus representing the problem of acoustic H-H AD has few things in common with the other three corpora and biases the results in Figure 4.12.

Analysing the other three corpora (VACC, SVC, and RBC) collected for acoustic H-M AD, we note that they have significantly more common features (139). Furthermore, these common functionals concern almost all the IS13_ComParE LLDs of all three categories (energy-, spectrum-, and voicing-related) together with the LLD deltas, meaning that, in our case, the feature selection is basically reduced to the selection of the optimal functionals for each LLD contour. The more advanced ComParE_LLD classifier, which is capable of learning its own, "custom" functionals for a specific

![Graph](image-url)

Fig. 4.11: Acoustic feature selection using RFE on all four corpora.
Fig. 4.12: Comparison of the optimal acoustic feature sets obtained with RFE on all four corpora. These sets correspond to the curves from Figure 4.11.

It can also be noted in Figure 4.11 that the optimal feature set size for SVC is greater than for the other three corpora. This difference was probably caused by the WOZ modelling of SVC dialogues as the WOZ setup did not seem convincing enough to some users, resulting in ambiguous addressee patterns that concerned a greater number of acoustic features.

4.4.3 Experiments with Various Acoustic Context Lengths

Next, we analyse the sensitivity of our ComParE_LLD classifier to acoustic context length variations. The amount of acoustic context was shown to be of great importance for audio-based emotion
recognition [Fedotov et al., 2018]. The authors of the latter study designed a model processing time-continuous acoustic features and tested it on a data set with various data sparsity coefficients. The coefficient $i$ means that each $i$-th time step of the time-continuous input is analysed, while all the remaining time steps are ignored. The model demonstrated a high degree of sensitivity to various acoustic context lengths and no sensitivity to various numbers of time steps if they cover the same temporal interval. We suppose that the choice of the optimal acoustic context length is also important for other paralinguistic problems besides audio-based emotion recognition.

The segmentation into context windows allows us to increase the number of training examples and hence partially offsets the lack of training data for ANNs. We suppose the most important acoustic addressee patterns to appear at short context lengths of around 1 second. Therefore, we use a logarithmic temporal scale depicted in Figure 4.13. We take a context length of 1 second as a reference point and then vary it by raising to different powers of two.

![Graph showing experiments with various acoustic context lengths on all four corpora.](image)

Fig. 4.13: Experiments with various acoustic context lengths on all four corpora. The performance curves are calculated using spline interpolation of exact measurements visualised as dots.
It is possible to align the obtained logarithmic scale with basic acoustic units. Given the mean syllable duration estimated by Greenberg [1999] for spontaneous English, we roughly assume the following alignment:

- 0 – 0.125 seconds: allophones;
- 0.125 – 0.5 seconds: syllables;
- 0.5 – 1 seconds: words;
- 1 – 2 seconds: collocations/syntagmas;
- 2 – 8 seconds: utterances.

In fact, these intervals may significantly overlap since syllable duration is known to be highly speaker-dependent [Greenberg et al., 2003]. German words and more complex acoustic units have longer durations compared to their English equivalents.

Context windows are extracted from audio files with an overlap of 75% of the window length. For each context length, the ComParE_LLD classifier is examined on the ten-fold LOSGO cross-validation. The resulting UAR is averaged over all folds.

All the curves from Figure 4.13 flatten beyond 2 seconds. Further context extensions do not contribute to acoustic AD. Moreover, there exists some performance loss beyond 2 seconds in every case. This loss can be explained in the following way: the model analysing longer context was trained on a smaller number of examples. In other words, the model was solving a more complicated task while having fewer training examples at its disposal. The acoustic context of 2 seconds was shown to be optimal for all four corpora, though the median values of their utterance length distributions depicted in Figure 4.14 – 4.17 vary considerably (see Table 4.3 for more detail). The

![Graph](image)

Fig. 4.14: Kernel density estimation of the VACC utterance length distribution for audio.
Fig. 4.15: Kernel density estimation of the SVC utterance length distribution for audio.

Fig. 4.16: Kernel density estimation of the RBC utterance length distribution for audio.
differences between the utterance length distributions arose due to the different SDS experimental setups and also due to the different domains of the systems. These factors resulted in various interaction complexity levels supported by the systems. It can also be noted in Figure 4.14 – 4.17 that the machine/child-directed utterance length distributions (dotted curves) always have smaller standard deviations compared to the human/adult-directed ones (dashed curves, see Table 4.3 for more detail). In other words, utterances directed to a special addressee tend to have a specific, relatively small amount of content. The aforementioned differences between the utterance length distributions become even more obvious if we measure the utterance durations not in seconds but in words (see Figure 4.5 – 4.8 and Table 4.1 for more detail).

According to our alignment, two-second speech fragments correspond to acoustic addressee patterns at the utterance level. This amount of context was also shown to be sufficient for audio-based H-M AD in English [Shriberg et al., 2013]. Comparing Table 4.1 and 4.3, we note that two seconds of speech equal slightly less than six words for the German corpora (VACC, SVC, and RBC) and slightly more than six words for the English corpus (HB).

### 4.4.4 Experiments with the ASR Metafeatures

It has already been shown by Tsai et al. [2015] that ASR metafeatures, particularly ASR confidence, are beneficial for audio-based H-M AD. The ASR confidence of machine-directed speech tends to be higher compared to human-directed speech since people try speaking more clearly while talking to a special addressee (machine). It is also interesting to check this approach on audio-based H-H AD. Using the Google Cloud ASR [Google LLC 2019] for German (on VACC, SVC, and RBC) and for English (on HB), we train our ASR_meta classifier on the training partitions from Table 4.2
Table 4.3: Properties of the utterance length distributions visualised in Figure 4.14 – 4.17. All values are given in seconds.

<table>
<thead>
<tr>
<th>Corpus name</th>
<th>Addressee label</th>
<th>Mode</th>
<th>Median</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VACC</td>
<td>Machine</td>
<td>0.800</td>
<td>1.390</td>
<td>1.818</td>
<td>1.323</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>0.660</td>
<td>1.344</td>
<td>1.815</td>
<td>1.569</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.750</td>
<td>1.371</td>
<td>1.817</td>
<td>1.408</td>
</tr>
<tr>
<td>SVC</td>
<td>Machine</td>
<td>3.250</td>
<td>3.930</td>
<td>4.950</td>
<td>3.274</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>3.500</td>
<td>5.232</td>
<td>6.385</td>
<td>4.511</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>3.330</td>
<td>4.404</td>
<td>5.680</td>
<td>4.017</td>
</tr>
<tr>
<td>RBC</td>
<td>Machine</td>
<td>1.400</td>
<td>2.192</td>
<td>2.693</td>
<td>2.014</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>1.000</td>
<td>2.006</td>
<td>2.619</td>
<td>2.175</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.250</td>
<td>2.141</td>
<td>2.667</td>
<td>2.072</td>
</tr>
<tr>
<td>HB</td>
<td>Child</td>
<td>1.100</td>
<td>1.190</td>
<td>1.494</td>
<td>0.797</td>
</tr>
<tr>
<td></td>
<td>Adult</td>
<td>1.100</td>
<td>1.280</td>
<td>1.723</td>
<td>1.209</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.100</td>
<td>1.220</td>
<td>1.582</td>
<td>0.983</td>
</tr>
</tbody>
</table>

and reach the following results. The UAR values given in Figure 4.18 were obtained on the test partitions from Table 4.2. As expected, the ASR metafeatures perform effectively for classical H-M AD on VACC and SVC. However, their performance drops dramatically for complexity-identical H-M AD on RBC and equals the chance level for H-H AD on HB. The latter result can be explained as follows. Adults’ sing-song intonations make their child-directed utterances difficult to recognise. Furthermore, adults’ speech often contains separate sounds and intonations and no verbal information when they talk to children. Therefore, the ASR confidence of child-directed utterances turns out to be unexpectedly low, even though they are directed to a special addressee. This fact makes the ASR metafeatures non-representative for audio-based H-H AD, though they are useful for audio-based H-M AD.

4.4.5 Excising the Trigger-Word from the VACC Acoustic Data

According to the results reported in the present chapter, VACC has turned out to be the easiest corpus for acoustic AD. Possible reasons for this are the absence of acoustic noise and the presence of the off-the-shelf SDS with real capabilities (SVC and RBC used WOZ simulations). Another reason is the trigger-word (“Alexa”) that users sometimes pronounce at the beginning of their requests to the system. The users do not have to use the trigger-word if the SDS is waiting for a response to its request within an ongoing dialogue or if the users have already pushed the PTT button of the Echo Dot device. Excising the trigger-word from the VACC audio data is problematic since it may
result in utterances no longer having a natural prosody. Furthermore, artefacts appearing after the data manipulation may give inadvertent clues to the ML system.

We employed the WebMAUS tool [Kisler et al., 2017] to locate trigger-word occurrences in audio files. After that, we excised these occurrences and trained our models on the obtained utterances. However, this data manipulation does not influence the classification results significantly. This outcome can be explained as follows. First, we do not use the ASR textual output directly for classification. Second, few of the machine-directed utterances in VACC start with the trigger-word. Third, the artefacts that arise after excising the trigger-word still have the same meaning for our models as this word itself.

4.5 Text Analysis

The most suitable corpus for text classification is SVC as it has already got manual transcripts [Batliner, Hacker, and Nöth, 2008]. The manual transcription of VACC and RBC was still in progress as of the period of writing the present thesis. The HB manual transcripts were unavailable to us neither. For VACC, RBC, and HB, we thus apply an off-the-shelf ASR [Google LLC 2019] to recognise users’ speech prior to text analysis. However, we do not want the difference between the ASR word recognition accuracy (which is between 85% and 90% for the corpora being considered) and the human listeners’ word recognition accuracy (which is close to 100%) to influence the results of our cross-corpus experiments. Therefore, we also use the ASR text output instead of the SVC man-
ual transcripts in order to make the classification performance comparable across all corpora. In the present chapter, all the transcripts are thus obtained applying the ASR. Less than 5% of all utterances from each of the corpora except HB, in which the ASR did not manage to recognise any words, are automatically classified as human-directed since it is a standard practice for SDSs to reject such utterances. 25% of all HB utterances, in which the ASR did not manage to recognise any words neither, are automatically classified as child-directed in accordance with their most frequent addressee label. More detailed experiments with textual AD on the SVC manual transcripts are presented in Chapter 6.

In general, we suppose VACC and SVC to be more appropriate for text-based AD than the other two corpora. The coupled human-directed and machine-directed RBC utterances have similar verbal content that complicates textual AD. A significant number of HB utterances has too much noise and/or no words (but only paralinguistic phenomena instead) to be recognised. These facts complicate textual AD on HB.

### 4.5.1 Cross-Corpus Experiments

Prior to conducting our cross-corpus experiments with text, we compare the domains of the considered corpora. We can do that directly comparing the vocabularies of the German corpora (VACC, SVC, and RBC). However, there exist many domain-irrelevant words, e.g., pronouns, prepositions, auxiliary verbs, etc., which are likely to get into the intersection of the vocabularies extracted from any corpora collected for the same language. In order to exclude domain-irrelevant words from consideration, we carry out stop-word filtering in combination with stemming. Various intersections of the vocabularies extracted from the considered corpora and subjected to stop-word filtering and stemming are depicted in Figure 4.19. The vocabulary sizes equal 886, 1113, and 606 words for VACC, SVC, and RBC respectively. The three vocabularies contain only 129 common words, i.e., most of the words in each vocabulary are unique. In other words, the domains of the considered

![Venn Diagram](image)

Fig. 4.19: Comparison of the vocabularies of the German corpora (VACC, SVC, and RBC) after stop-word filtering and stemming.
corpora have only minor similarities. This fact complicates any applications of models with a fixed vocabulary size, e.g., \( n \)-gram-based LMs, to cross-corpus experiments on the considered corpora. For this reason, we employ the Sem.BLSTM classifier (Figure 4.4) that uses distributed word representations.

Similarly to Section 4.4.1, we conduct a series of cross-corpus experiments with the Sem.BLSTM classifier, applying the same KOCI scheme. Additionally, we perform LOCO experiments using the latter model.

The Sem.BLSTM classifier is trained and tested on the data partitions from Table 4.2. Results of KOCI experiments on all four corpora are visualised in Figure 4.20a. Including HB into the KOCI experiments does not make much sense as this corpus uses English word embeddings which are not expected to be compatible with German word embeddings employed for the other corpora, although both groups of word embeddings have the same dimensionality of the word vectors. However, we keep the HB data for consistency.

Let us denote the UAR values from Figure 4.20a as a matrix \( A^* \) with elements \( a^*_{train\_corpus, test\_corpus} \). The obtained cross-corpus relationships for the textual modality are pretty similar to those depicted in Figure 4.9 for the acoustic modality. First of all, we note that \( a^*_{VACC,VACC} \) and \( a^*_{SVC,SVC} \) are significantly greater than the other elements of \( A^* \). In other words, the textual AD problems of VACC and SVC turned out to be essentially easier for our Sem.BLSTM classifier than those of the other two corpora. In addition, \( a^*_{VACC,SVC} \) and \( a^*_{SVC,VACC} \) are significantly greater than the other off-diagonal elements of \( A^* \). The latter means a clear relationship between VACC and SVC in the verbal sense despite the essential difference between the domains of these corpora. Therefore, this relationship is likely to be connected with the speaking style and syntax rather than with individual lexical units.

The role of syntactical features for AD is thoroughly studied in Chapter 6.

![Fig. 4.20: Cross-corpus experiments with the Sem.BLSTM classifier on various corpora. All results are presented in terms of UAR.](image)
Even though RBC is the most challenging corpus for the text classifier, this corpus also demonstrates some relationship with VACC: $a_{RBC,VACC}^*$ is noticeably above a chance performance of 0.5 and also above $a_{RBC,RBC}^*$. At the same time, the symmetrical element $a_{VACC,RBC}^*$ is just slightly above the chance level. In other words, the model trained on RBC performs better on VACC than on the RBC test set. At first sight, it seems to be a contradiction. To resolve it, we need to recall the RBC experimental setup described in Section 3.3. The addressee patterns captured within RBC are extremely subtle since visual and contextual differences between a human addressee and a machine were minimised for the user. This gives us the following explanation for the contradiction arisen: the training on the complexity-identical H-M AD setup of RBC allows the model to learn true verbal addressee patterns that are also typical for the classical H-M AD setup of VACC. However, the opposite statement does not work due to collateral factors, such as differences between the human addressee’s appearance and the SDS appearance, between the domains and complexity levels of human-directed and machine-directed utterances, etc. These factors influence the user’s verbal behaviour within the VACC setup, but not within the RBC setup, and cause the model to overfit. SVC and RBC do not exhibit such a relationship probably due to the differences between the SVC and RBC experimental setups; the former setup supports verbose, complex machine-directed utterances, while the latter setup accepts shorter, simpler machine-directed utterances. RBC and its relationship with VACC and SVC are thoroughly investigated in Section 5.5.

4.5.2 Experiments with Various Verbal Context Lengths

We wonder how early it is possible to predict the addressee within one utterance. The VACC, SVC, RBC, and HB utterance length distributions depicted in Figure 4.5 – 4.8 have median values of 4, 9, 5, and 4 words respectively (see Table 4.1). However, the trends illustrated in Figure 4.21 demonstrate that the main performance gain for each corpus is reached with a context length of just the first six words; further context extensions contribute much less to the AD performance. The resulting performance values have been computed on the data partitions from Table 4.2. Comparing Table 4.3 and 4.1, we note that six words approximately equal two seconds of speech that confirms the results obtained in Section 4.4.3.

It can also be noted in Figure 4.5 – 4.8 that the machine/child-directed utterance length distributions (dotted curves) always have smaller standard deviations compared to the human/adult-directed ones (dashed curves, see Table 4.1 for more detail). In other words, utterances directed to a special addressee tend to have a specific, relatively small amount of content.
4.5.3 Excising the Trigger-Word from the VACC Textual Data

The measurements marked with crosses in Figure 4.21 denote experiments on a different version of the VACC data. We obtained this version of VACC by removing the trigger-word ("Alexa") to complicate the text classification task for the Sem_BLSTM model. In contrast to excising the trigger-word from the acoustic data in Section 4.4.5, we can seamlessly excise the trigger-word from text without damaging the text naturalness. This word tends to appear at the beginning of some utterances. As a result, the classification performance of the Sem_BLSTM model slightly drops at context lengths of below 6 words compared to the original version of the VACC data, i.e., the classifier requires a longer context to determine the addressee of an utterance in the absence of the trigger-word. However, the performance drop becomes insignificant as we extend the context being analysed by the model. We can conclude that the trigger-word does not play a crucial role for H-M AD on VACC.

4.6 Summary

According to our experiments introduced in the present chapter, the AD problem has turned out to be extremely corpus-dependent. Placing our corpora in the order of their descending UAR values, we obtain the following sequence: VACC, SVC, HB, and finally RBC. This sequence is identical for both acoustic and textual modalities and can be clearly observed in Figure 4.13 and 4.21. VACC is
thus the easiest corpus for our classifiers. There exist several explanations for this result. Firstly and most importantly, the VACC experimental setup includes a real, off-the-shelf SDS (Amazon Alexa Echo Dot) that supports relatively simple requests in specific domains. In contrast to this SDS, the technical systems employed in the SVC and RBC experimental setups were WOZ-simulated, and therefore these systems could seem unconvincing to some users, resulting in ambiguous addressee patterns for speech and text. Secondly, the VACC SDS has a specific appearance (physical desktop device) that differs from the appearance of the human companion sitting beside. The two aforementioned factors make the user perceive the technical system and the human interlocutor differently, resulting in obvious acoustic and textual differences between machine-directed and human-directed utterances. Thirdly, the VACC speech data, which has been recorded indoors, contains a much smaller amount of background noise compared to SVC and HB, some sessions of which have been collected outdoors.

There exists a clear relationship between VACC and SVC in the sense of both acoustic and textual addressee patterns (see Figure 4.9 and 4.20a for more detail). This relationship can be explained by the similar experimental setups of these two corpora.

The main reasons for such a low AD performance on HB are a large amount of background noise and adults’ atypical spoken behaviour.

Despite the absence of background noise, RBC was shown to be the most challenging corpus for both acoustic and textual AD. There exist several explanations for this. First, the RBC experimental setup brings the conversational role of the SDS as close as possible to the human interlocutor’s conversational role. As a result, the H-M and the H-H conversation have minimal differences between their domains, complexity levels, and lexical content. Second, both the SDS and the human conversational partner have no physical appearance; the user can not see them during a phone call, and therefore he/she is not subject to the effect of a visible counterpart. Third, the user does not actually need to emphasise the desirable addressee as he/she is talking to a single conversational partner throughout a phone call. Apparently, the latter pragmatical aspect plays an important role in the process of emphasising addressees by means of speech.

Although RBC is the most challenging corpus for AD, the RBC textual data can be effectively augmented using VACC and SVC. See Figure 4.20b for more detail.

The set of the representative acoustic functionals is highly corpus-dependent. The four considered corpora have few representative acoustic functionals in common. These functionals concern spectral and cepstral LLDs used to identify the notes, pitch, rhythm, and melody. See Section 4.4.2 for more detail.

Two-second speech fragments, which contain around six words, are sufficient for reliable acoustic and textual AD on all four corpora. These fragments correspond to spoken addressee patterns at the utterance level. Further context extensions improve the AD performance insignificantly. See Section 4.4.3 and 4.5.2 for more detail.

As expected, the ASR metafeatures are relevant to H-M AD. However, they turn out to be completely useless for H-H AD due to adults’ sing-song intonations that make their child-directed utterances difficult to recognise. See Section 4.4.4 for more detail.

Excising the trigger-word ("Alexa") from the VACC acoustic data does not influence the AD performance for the following reasons. First, we do not use the ASR textual output directly for classification. Second, few of the machine-directed utterances in VACC start with the trigger-word.
Third, the artefacts that arise after excising the trigger-word still have the same meaning for our models as this word itself. However, excising the trigger-word from the VACC textual data does decrease the AD performance for short context lengths (below six words) as this word usually appears at the beginning of some utterances. The performance drop becomes insignificant if we extend the context being analysed by our text classifier (see Section 4.5.3 for more detail).

Some of the methods and results reported in the present chapter have been published in [Akhtiamov, Siegert, et al., 2019; Akhtiamov et al., 2020].
AD is a modern paralinguistic and dialogue challenge that especially arises in voice assistants. In the present chapter, we distinguish addressees, using speech classification for the following problem statements: classical H-M AD (represented by VACC and SVC), H-H AD (represented by HB), and complexity-identical H-M AD (represented by RBC, see Section 1.2 for more detail on the problem statements). We compare the problem statements pairwise: classical H-M AD versus H-H AD and classical H-M AD versus complexity-identical H-M AD. In each case, we jointly solve both classification problems, employing various models, including several DNN architectures, for speech and text analysis. Furthermore, we investigate how different corpora influence each other, introducing a novel mixup-based approach to cross-corpus data augmentation. Our mixup-based method, which merges acoustic data from different sources, is shown to be beneficial for DNN regularisation. This method allows us to reach a higher classification performance on H-M AD and also to train a multipurpose DNN that is capable of solving both H-M AD and H-H AD problems simultaneously. Moreover, we introduce new competitive baselines for VACC and RBC.

The present chapter is organised as follows. We propose several audio classifiers in Section 5.1 and also introduce a novel data augmentation technique in Section 5.2. Section 5.3 is dedicated to an additional metric for classification performance. In Section 5.4, we jointly study classical H-M AD and H-H AD. Classical H-M AD and complexity-identical H-M AD are jointly studied in Section 5.5. Section 5.6 is dedicated to the fusion of speech and text for AD. The content of the present chapter is summarised in Section 5.7.

5.1 Proposed Classifiers

In the present chapter, we employ the four classifiers proposed in Section 4.1: ComParE_func, ComParE_LLD, ASR_meta, and Sem_BLSTM. In addition to them, we introduce an end-to-end model (E2E) that performs raw speech classification. All together, these classifiers, except ASR_meta and Sem_BLSTM, satisfy a uniform concept formulated in Section 4.1. According to this concept, a more complex model should be applied to process lower-level input feature representations and vice versa.

Most existing studies on acoustic H-M AD [Batliner, Hacker, and Nöth, 2008; Shriberg et al., 2013; Tsai et al., 2015] are focused on acoustic feature design rather than on model design. Meanwhile, any additional data preprocessing inevitably results in information losses that should be
avoided. Therefore, data-driven and particularly end-to-end models based on DNNs dealing with low-level features can improve acoustic AD. Such models are capable of performing feature selection and feature transformation implicitly for a particular problem. For this purpose, Mallidi et al. [2018] designed a huge DNN comprising multiple layers of LSTM units. In their next work [C.-W. Huang et al., 2019], the researchers investigated this model on different dialogue types. The best performing network configuration had around 16 million parameters and was trained on 250 hours of real speech data. However, neither this network nor that one proposed by Norouzian et al. [2019] for a similar task were end-to-end since they both were receiving log filter-bank features as the input for acoustic H-M AD. In addition to handcrafted features (IS13_ComParE LLDs and their functionals), we want our model to learn its own feature representations directly from raw waveforms. We compensate the lack of training data for our DNNs, carefully choosing the model architecture and applying the proposed mixup-based method to cross-corpus data augmentation.

5.1.1 End-to-End Classifier

The proposed classifier (E2E) performs end-to-end speech signal processing. In fact, it is a more advanced version of the ComParE_LLD model from Section 4.1.2, where the LLD extractor is replaced by a CNN component. The input segmentation procedure, the RNN, and the postprocessing component remain the same. The CNN component consists of a stack of one-dimensional convolutional layers and pooling layers that perform feature extraction from raw audio. See [LeCun et al., 1998] and Section 2.4.4 for more detail on CNNs. The convolution is conducted over the temporal dimension of the speech signal. Rectified Linear Unit (ReLU) is used as an activation function for each convolutional layer. A batch normalisation operator is applied between each convolution and activation. The abstract features returned by the CNN component represent some speech characteristics that the network "considers" to be important for the classification problem being solved. These features are fed to an RNN component that is identical to that one of the ComParE_LLD classifier. As a result, we obtain a Convolutional Recurrent Neural Network (CRNN) that is quite similar to the one suggested by Trigeorgis et al. [2016] for acoustic emotion recognition. However, the initial network architecture specified in the latter study showed the lowest results among all models (including much simpler ones) on the H-H AD problem of HB probably due to a lack of perceptive abilities [Schuller et al., 2017]. For this reason, we replaced the initial two-layer CNN component by a deeper one. We took the five-layer SoundNet architecture [Aytar et al., 2016] as the reference point for our CNN, cut off its last convolutional layer, scaled the filter sizes in accordance with our input signal resolution and also scaled the number of units in each layer in accordance with the available amount of our training data. Although a sampling rate of 8 kHz is considered to be sufficient for most speech applications, we decided to leave the original sampling rate of our data (16 kHz) as just a small acceleration of computations and no significant difference in the classification performance were spotted after the downsampling. The final shape of the E2E classifier is depicted in Figure 5.1.
5.1 Proposed Classifiers

Fig. 5.1: E2E classifier. The architecture of the RNN component is visualised in Figure 4.2.
As the result of a trial-and-error procedure, we have chosen the following hyperparameters for the E2E classifier: a CNN component containing two layers of 128 LSTM units each, Gaussian noise applied to the input signal if mixup is deactivated, a dropout of 20% if mixup is deactivated, cross-entropy as a loss function, Adam [Kingma and Ba, 2014] as a weight optimisation algorithm, 100 epochs, and a mini-batch size of 32 examples. The initial learning rate is selected from the set \( \{10^{-3}, 10^{-4}, 10^{-5}\} \) and then divided by 10 if there is no performance improvement observed for the past 10 epochs on the development set. We make checkpoints, saving the current model weights at each epoch and using the best checkpoint as the resulting model according to its performance on the development set.

It is important to note that each of our DNN-based classifiers (ComParE_LLD and E2E) has a considerably smaller number of learnable parameters (0.3 million) compared to the network proposed by Mallidi et al. [2018] (16 million).

5.1.2 AP_meta Classifier

In the present chapter, we want to leverage the benefits of all four models proposed in Section 4.1 and 5.1: ComParE_func, ComParE_LLD, E2E, and ASR_meta. Besides the individual classifiers, we introduce an acoustic-prosodic metaclassifier (AP_meta) based on the fusion of their confidence scores.

The concept of the AP_meta classifier is similar to that one visualised in Figure 2.14d. The confidence scores of the individual models are concatenated and fed to an SVM with a linear kernel as shown in Figure 5.2. The metamodel is trained on a development set, while the low-level classifiers are trained on a training set. According to our preliminary experiments, the metaclassification tends to perform more effectively than a simple score fusion.

5.2 Proposed Approach to Data Augmentation

Overfitting is a critical issue for ANNs and especially for DNNs that possess large numbers of learnable parameters. Data augmentation and model regularisation help us to combat this problem. In our study, we use several basic regularisation techniques, such as a large learning rate, Gaussian noise, dropout, and mixup.

A high learning rate at the beginning of the training process is a trivial way to reduce overfitting. Furthermore, it can speed up the training. However, a too large learning rate may result in underfitting.

Adding Gaussian noise to the input signal is another popular way to reduce overfitting. On the one hand, this method brings certain variability to the input data. On the other hand, the amount of noise should be carefully chosen since too much noise may lead to underfitting.

The dropout method proposed by Srivastava et al. [2014] is meant to be used in DNNs. A certain percentage of the connections between the neighbouring layers is randomly removed for each training example. A large dropout ratio may result in underfitting.
Fig. 5.2: AP_meta classifier including four low-level models. The detailed architectures of the ComParE_func, the ComParE_LLD, the E2E, and the ASR_meta classifier are given in Figure 4.1, 4.2, 5.1, and 4.3 respectively.
Training our DNNs, we apply a simple yet efficient approach called mixup [H. Zhang et al., 2017]. This method regularises our model by encouraging it to behave linearly in the vector space between seen data points. The method generates artificial examples in form of \((X, Y)\) as linear combinations of the feature and label vectors taken from two arbitrary real examples \((X_p, Y_p)\) and \((X_q, Y_q)\) and mixed at a proportion \(\lambda\) in the following way:

\[
X = \lambda X_p + (1 - \lambda) X_q, \quad (5.1)
\]
\[
Y = \lambda Y_p + (1 - \lambda) Y_q, \quad (5.2)
\]

where \(\lambda\) is randomly generated from a \(\beta\)-distribution for each artificial example. This distribution is defined as follows by a coefficient \(\alpha\) that lies within the interval \((0, \infty)\) and determines the probability that our generated example lies close to one of real examples:

\[
f(x; \alpha) = \frac{1}{B(\alpha, \alpha)} x^{\alpha - 1} (1 - x)^{\alpha - 1}, \quad (5.3)
\]

where \(B(\alpha, \alpha)\) is a normalisation constant (beta function) to ensure that the total probability equals 1. The closer to zero the \(\alpha\) value is, the more the generated examples bias toward the real ones, and vice versa.

We introduce a novel Algorithm 2 that combines mixup and multitask learning strategies and allows our model to learn data from several similar corpora in a smooth manner [Akhtiamov, Siegert, et al., 2019].

An important feature of our algorithm is connected with target vectors of artificial examples generated with the algorithm. If \(Y_p\) and \(Y_q\) from Formula 5.2 are different hard targets (one-hot vectors) of a classification problem, \(Y\) will be a soft target. This solution provides better model regularisation and generalisation over various classes and thereby partially resolves the imbalanced data problem which was formulated in Section 4.3.

Even though the parameter \(\alpha\) of the \(\beta\)-distribution is mathematically defined within the interval \((0, \infty)\), we noted that the proposed algorithm gave adequate results with \(\alpha\) values lying in the interval \((0, 1)\), i.e., when the algorithm biased toward real examples and thereby generated more realistic artificial ones. In particular, \(\alpha = 0.5\) provides effective results in most algorithm launches. In most of the mixup applications investigated by H. Zhang et al. [2017], \(\alpha\) was lying within the interval \([0.1, 0.5]\) that also confirms our observations. Greater \(\alpha\) values may lead to underfitting. The probability density function of the \(\beta\)-distribution with \(\alpha = 0.5\) is depicted in Figure 5.3.

Another important parameter of the proposed algorithm is a coefficient \(k\) that defines the proportion of the number of artificial examples that should be generated and the number of real examples. Suppose we want to use our Algorithm 2 to merge \(n\) different corpora of real data. The proposed algorithm thus increases the number of training examples in \(n(k + 1)\) times if mixup is activated. According to our experiments, the algorithm demonstrates a positive effect already at \(k = 2\).

Algorithm 2 mixes real examples regardless of their class and corpus. According to our experiments, mixing only those examples belonging to one class or to one corpus does not improve the classification performance in comparison with the current version of the algorithm.
The proposed algorithm does not result in any considerable delays during the training process since artificial examples are generated batchwise. The parameters $\alpha$ and $k$ are fine-tuned on a development set.

**input**: target corpus $T$, auxiliary corpora $C_1,...,C_n$, mini-batch size $l$, parameter $\alpha$, proportion of the number of artificial examples and the number of real examples $k$.

**output**: training mini-batches.

**Algorithm 2**: One epoch of the mixup-based training procedure.
It is important to note the compatibility of Algorithm 2 with various loss functions and activation functions for the output layer. The most straightforward way to apply mixup for classification tasks is to use it in combination with the Mean Square Error (MSE) loss function:

$$L_{MSE} = \frac{\sum_{i=1}^{|C|} (y_i - y_i^{(p)})^2}{|C|},$$

(5.4)

where $y_i$ and $y_i^{(p)}$ are a model target and a prediction for the $i$-th class respectively. If we employ the ReLU activation function in the output dense layer, the values $y_i^{(p)}$ can be computed as follows:

$$y_i^{(p)} = \max(0, z_i),$$

(5.5)

where $z_i$ is a weighted sum of all inputs for the $i$-th neuron. The classification task is thus transformed into a regression task in order to fully explore the potential of soft targets. Such artificial targets appear as the result of applying mixup to real data points belonging to different classes. The model thus learns to predict the proportion of each class in the mixed input signal.

However, it is also possible to combine Algorithm 2 with classification loss functions, e.g., cross-entropy which is also known as log loss:

$$L_{CE} = -\sum_{i=1}^{|C|} y_i \log(y_i^{(p)}),$$

(5.6)

where $y_i$ is a model target rounded to the nearest integer. If we employ the softmax activation function in the output dense layer, the values $y_i^{(p)}$ can be computed as follows:

$$y_i^{(p)} = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, ..., K \text{ and } z = (z_1, ..., z_k) \in \mathbb{R}^K,$$

(5.7)

where $z_i$ is a weighted sum of all inputs for the $i$-th neuron.
The model thus learns to predict the dominating class in the mixed input signal. This is reasonable given the probability density function depicted in Figure 5.3. For $\alpha < 1$, this function causes the artificial targets to bias to one of the real targets which are one-hot vectors.

According to our preliminary experiments, the loss functions defined in Formula 5.4 and 5.6 show very similar classification performance values on our two-class problem, while the latter loss function provides a noticeably faster training. We therefore employ cross-entropy as the loss function and softmax as the activation function of the output layer for all our ANNs with and also without using Algorithm 2. We also round every obtained artificial target to the nearest hard target (one-hot vector), if Algorithm 2 is applied.

5.3 Metrics

Besides the UAR metric defined in Section 4.2, we additionally use Unweighted Average Precision (UAP) to estimate the classification performance on RBC in Section 5.5.3. According to our experiments reported in Chapter 4, this corpus turned out to be the most challenging for acoustic AD compared to the others. The second metric ensures the credibility of the results obtained, although the reached classification improvement is relatively small. Siegert et al. [2019] also assessed the classification performance on RBC in terms of both UAR and UAP.

In ML, the precision of a class is the ratio of the number of the correctly classified data points belonging to the class to the total number of the data points assigned to this class. The formal definition of precision is as follows:

$$P_c = \frac{|A_c \cap B_c|}{|B_c|},$$

where $C$ is a set of all classes, $c$ is the class for which we calculate the precision value $P_c$, $A_c$ is the set of all data points which actually belong to the class $c$, and $B_c$ is the set of the data points classified as belonging to the class $c$. There exist several approaches to averaging the precision values of individual classes. Unweighted average precision means a simple averaging that can be formally described as follows:

$$P_{UAP} = \frac{\sum_{c \in C} P_c}{|C|}. \quad (5.9)$$

5.4 Experiments on the HomeBank Corpus

In this section, we jointly study classical H-M AD (represented by VACC and SVC) and H-H AD (represented by HB), using multitask learning and mixup.

5.4.1 Cross-Corpus and Multitask Classification Using Simple Models

We conduct two series of cross-corpus and multitask experiments with the ComParE_func model, applying an KOCl and a LOCO scheme as specified in Section 4.4.1. This time, we confine to
three corpora: VACC, SVC, and HB. The classifier is trained and tested on the data partitions from Table 4.2. Within these experiments, we do not perform feature selection and do not use mixup. Results of the two experimental series are visualised in Figure 5.4. Let us denote the matrix from Figure 5.4a as $\tilde{A}$, its element as $\tilde{a}_{\text{train \\ corpus, test \\ corpus}}$, the matrix from Figure 5.4b as $\tilde{B}$, and its element as $\tilde{b}_{\text{corpus \\ excluded \\ from \\ train, test \\ corpus}}$. $\tilde{a}_{\text{VACC, SVC}}$ and $\tilde{a}_{\text{SVC, VACC}}$ are considerably greater than the other off-diagonal elements of $\tilde{A}$, demonstrating a clear relationship between VACC and SVC. This result motivates us to explore the potential of the cross-corpus data augmentation on these two corpora by means of mixup and deep learning in Section 5.4.3.

![Figure 5.4: Preliminary cross-corpus experiments with the ComParE_func classifier on VACC, SVC, and HB. All results are presented in terms of UAR.](image)

The matrix $\tilde{A}$ does not reveal any relationship between HB and the other two corpora, though an interesting trend can be noted in $\tilde{B}$. $\tilde{b}_{\text{SVC, VACC}}$ and $\tilde{b}_{\text{HB, VACC}}$ are similar to $\tilde{a}_{\text{VACC, VACC}}, \tilde{b}_{\text{VACC, SVC}}$ and $\tilde{b}_{\text{HB, SVC}}$ are close to $\tilde{a}_{\text{SVC, SVC}},$ and $\tilde{b}_{\text{VACC, HB}}$ and $\tilde{b}_{\text{SVC, HB}}$ are similar to $\tilde{a}_{\text{HB, HB}}$. Altogether, these results mean that a single ComParE_func model trained on examples from two arbitrary corpora demonstrates an adequate performance on them both as if the model were trained on each corpus separately or, in other words, that the three classification problems represented by the different corpora are non-contradictory. We can also note that $\tilde{b}_{\text{VACC, HB}} > \tilde{b}_{\text{SVC, HB}}$, meaning that VACC influences the HB data more than SVC does. A possible explanation for this is a strong correlation between the features and labels of VACC, which was shown to be the easiest corpus for the ComParE_func classifier (see Figure 4.9 for more detail). This corpus probably caused the model to overfit, reducing its classification performance on HB. This hypothesis is verified in Section 5.4.4.

### 5.4.2 Feature Selection

Next, we want to determine the degree of similarity between the considered corpora in the sense of common relevant acoustic features. For this purpose, we apply RFE in combination with the Com-
ParE_func model similarly to the experiments from Section 4.4.2. The coefficients of the normal vector of the linear SVM are used as attribute weights. Figure 5.5 demonstrates RFE curves obtained by applying ten-fold LOSGO on various corpora. The resulting performance is calculated as UAR averaged over all folds for each reduced feature set. One set of curves visualises RFE on individual corpora. The curves from this set are identical to those depicted in Figure 4.11. Another set of curves marked with an asterisk reflects joint RFE across all three corpora. In order to obtain the latter curves, we compute a single weighting list on all the corpora merged in equal proportions and use a single ComParE_func model, which is also trained on all three corpora, to calculate the classification performance on them. We consider a feature set to be optimal if further RFE iterations worsen the classification performance stably.

Intersections of three optimal feature sets, each of which was obtained on a random fold with RFE on individual corpora, are depicted in Figure 5.6. The representative acoustic functionals vary considerably: VACC, SVC, and HB have only 1170, 2020, and 1400 relevant features out of 6373 respectively, while having only 52 attributes in common.

The underlying LLDs of these 52 common functionals are as follows:

- Energy-related, prosodic: audspec lengthL1norm sma, audspec lengthL1norm sma de, pcm zcr sma;
- Spectrum-related, spectral: audSpec Rfilt sma, audSpec Rfilt sma de, pcm fftMag spectralRollOff25.0 sma, pcm fftMag spectralRollOff25.0 sma de, pcm fftMag spectralVariance sma;
- Spectrum-related, cepstral: mfcc sma, mfcc sma de;

![Fig. 5.5: Acoustic feature selection using RFE on VACC, SVC, and HB.](image-url)
Fig. 5.6: Comparison of the optimal acoustic feature sets obtained with RFE on VACC, SVC, and HB. These sets correspond to the curves from the first row of the legend (without an asterisk) in Figure 5.5.

- Voicing-related, prosodic: F0final sma;
- Voicing-related, voice quality: voicingFinalUnclipped sma.

The IS13_ComParE feature notations can be found in [Eyben, 2015] and Section 2.2.1. It is hard to say anything specific about the common functionals and also about the results of the joint RFE (see the curves marked with an asterisk in Figure 5.5). The only evident observation is that the merged corpora bring slight noise to each other’s data regardless of the selected features. Let us denote one set from Figure 5.6 as $S_{\text{corpus\_name}}$. It can be seen in Figure 5.5 that an adequate joint classification performance is reached at a feature set size of 4600. In other cases, the joint RFE results in a significant performance loss at least on one corpus. As expected, this feature set contains all functionals from $S_{\text{VACC}} \cap S_{\text{SVC}}$, $S_{\text{VACC}} \cap S_{\text{HB}}$, $S_{\text{SVC}} \cap S_{\text{HB}}$, and other features.

5.4.3 Acoustic Context Length Variations and Mixup

All the experiments below are presented in terms of UAR for comparability with the existing studies. All statistical comparisons are drawn applying a $t$-test with a significance level of 0.05. In the present section, we extend the experiments from Section 4.4.3, applying the proposed Algorithm 2 in combination with our DNN-based classifiers to various acoustic context windows. Acoustic context length was shown by Fedotov et al. [2018] to be an important hyperparameter for paralinguistic problems. We take a context window length of 1 second as a reference point and then vary it by raising to different powers of two. The context windows are cut out of audio files with an overlap of 75%. The idea behind this preprocessing is explained in Section 4.4.3.

Performance curves obtained with the ComParE_LLD classifier on ten-fold LOSGO are depicted in Figure 5.7 – 5.9. In each case, the training set of the target corpus (on the test set of which the resulting UAR values are computed) is utilised unchanged ($target\_corpus$), mixed
with itself \((mix(target_corpus))\), or mixed with itself and with the training set of another corpus \((mix(target_corpus, auxiliary_corpus))\). The resulting UAR values are averaged over all folds. The dots connected using spline interpolation denote exact measurements.

We noted that applying several regularisation techniques, e.g., mixup, Gaussian noise, and high dropout ratios, at once eliminated their positive effect, and therefore we use small dropout ratios and no additional noise when Algorithm 2 is activated.

The dashed curve is located above the solid one in all three cases, i.e., Algorithm 2 results in a significant performance improvement already when applied to the same corpus. Adding another corpus to the mixup procedure influences the performance depending on a context window length. Mix(VACC, SVC) significantly surpasses mix(VACC) on VACC for a context window of 2 seconds. Mix(SVC, VACC) significantly outperforms mix(SVC) on SVC for a context window of 0.5 seconds. A possible explanation for these two results is that SVC has generally longer utterances (see Table 4.3 for more detail) and probably longer acoustic addressee patterns compared to VACC. Mix(HB) does not benefit from adding another corpus to the mixup procedure.

The curves from Figure 5.7 flatten beyond 0.5 seconds, meaning that VACC is less sensitive to context length variations than SVC and HB. The optimal context length, which provides the highest UAR, is 2 seconds for VACC and SVC and 1 second for HB. However, the latter corpus demonstrates virtually the same result at a longer context window of 2 seconds. The difference between the performance values obtained on HB at 1 and 2 seconds is thus statistically insignificant. The E2E classifier shows a similar behaviour on various context windows and reaches the highest UAR at the same context window of 2 seconds on all three corpora. This fact motivates us to confine to a single context window length of 2 seconds in our future experiments within the present chapter.

Fig. 5.7: Classification performance of the ComParE_LLD model combined with Algorithm 2 on VACC.
Fig. 5.8: Classification performance of the ComParE_LLD model combined with Algorithm 2 on SVC.

Fig. 5.9: Classification performance of the ComParE_LLD model combined with Algorithm 2 on HB.
These results confirm our conclusion regarding the optimal context length from Section 4.4.3. They also confirm an earlier conclusion drawn by Shriberg et al. [2013] regarding the optimal context length for acoustic H-M AD in English.

Table 5.1 contains the exact UAR values of the two-second performance slices for both neural networks. Similarly to the results presented in Figure 5.7 – 5.9, the values from Table 5.1 are obtained on ten-fold LOSGO and averaged over all folds. The ComParE_LLD model demonstrates a higher performance compared to the E2E model overall, except HB, on which both classifiers behave similarly. In contrast to the ComParE_LLD model, the E2E classifier does not benefit from Algorithm 2. This result contradicts the supposition made by H. Zhang et al. [2017] to apply mixup to raw speech data and may be naturally explained in the following way: after applying mixup to raw speech signals, our augmented data sounds like crowd noise that confuses the E2E model being unable to handle the cocktail party effect. However, this is not the case for some handcrafted features, e.g., logarithmic attributes, as applying mixup to them does not necessarily mean a simple overlapping of two waveforms, from which these features were extracted. Another possible explanation is trivial: the E2E classifier requires a larger amount of real training data so that Algorithm 2 can exhibit a positive effect. We conclude that applying mixup makes more sense for acoustic features of a higher abstraction level than raw data, e.g., handcrafted LLDs or features extracted with a CNN. In the present study, we confine to two extreme cases: LLDs and raw waveforms.

Table 5.1: Two-second UAR slices. Each marker corresponds to a curve of the same style in Figure 5.7 – 5.9. The greatest value in each row is marked in bold.

<table>
<thead>
<tr>
<th>Test corpus</th>
<th>Classifier</th>
<th>———</th>
<th>———</th>
<th>———</th>
<th>———</th>
<th>mix(all)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VACC</td>
<td>ComParE_LLD</td>
<td>0.879</td>
<td>0.890</td>
<td>0.901</td>
<td>0.873</td>
<td>0.886</td>
</tr>
<tr>
<td></td>
<td>E2E</td>
<td>0.853</td>
<td>0.834</td>
<td>0.852</td>
<td>0.845</td>
<td>0.846</td>
</tr>
<tr>
<td>SVC</td>
<td>ComParE_LLD</td>
<td>0.813</td>
<td>0.823</td>
<td>0.804</td>
<td>0.795</td>
<td>0.818</td>
</tr>
<tr>
<td></td>
<td>E2E</td>
<td>0.764</td>
<td>0.756</td>
<td>0.758</td>
<td>0.749</td>
<td>0.761</td>
</tr>
<tr>
<td>HB</td>
<td>ComParE_LLD</td>
<td>0.631</td>
<td>0.645</td>
<td>0.627</td>
<td>0.640</td>
<td>0.636</td>
</tr>
<tr>
<td></td>
<td>E2E</td>
<td>0.647</td>
<td>0.632</td>
<td>0.633</td>
<td>0.616</td>
<td>0.631</td>
</tr>
</tbody>
</table>

5.4.4 Cross-Corpus and Multitask Classification Using Deep Neural Networks

The experiments reported in the present section are conducted on the data partitions specified in Table 4.2. Six series of cross-corpus experiments using the ComParE_LLD and the E2E classifier are visualised as performance matrices in Figure 5.10. Let us denote the matrix from Figure 5.10a as $\tilde{A}$ and its element as $\tilde{a}_{row\_name, column\_name}$, the matrix from Figure 5.10b as $\tilde{B}$ and its element as $\tilde{b}_{row\_name, column\_name}$, etc.
Fig. 5.10: Cross-corpus experiments with the ComParE_LLD and the E2E classifier on VACC, SVC, and HB. The "+mix" mark means that Algorithm 2 is activated. All results are presented in terms of UAR. The acoustic context window length being used equals 2 seconds.
The matrices $\tilde{A}$ and $\tilde{B}$ show results of KOCI experiments on the ComParE_LLD model combined with Algorithm 2 and on the E2E model without mixup respectively. As we know from Section 5.4.3, the E2E classifier does not benefit from Algorithm 2. $\tilde{a}_{VACC,SVC}$ and $\tilde{a}_{SVC,VACC}$ are considerably greater than the other off-diagonal elements of $\tilde{A}$. $\tilde{b}_{VACC,SVC}$ and $\tilde{b}_{SVC,VACC}$ are also significantly greater than the other off-diagonal elements of $\tilde{B}$. Similarly to the matrix $\tilde{A}$ from Figure 5.4a, these two results demonstrate a clear relationship between VACC and SVC, which was detected by all of our classifiers, especially by the E2E model.

The other four matrices from Figure 5.10 contain results of LOCO experiments: $\tilde{C}$ and $\tilde{D}$ – without mixup, $\tilde{E}$ and $\tilde{F}$ – with Algorithm 2. The elements $\tilde{c}_{VACC,HB}$, $\tilde{c}_{SVC,HB}$, $\tilde{d}_{VACC,HB}$, and $\tilde{d}_{SVC,HB}$ are close to a random-choice UAR of 0.5, meaning that both neural networks perceive HB as noise and completely ignore it in favour of another corpus. However, the situation changes if we apply the proposed Algorithm 2; the elements $\tilde{e}_{VACC,HB}$ and $\tilde{e}_{SVC,HB}$ are similar to $\tilde{a}_{HB,HB}$ as well as the elements $\tilde{f}_{VACC,HB}$ and $\tilde{f}_{SVC,HB}$ being close to $\tilde{b}_{HB,HB}$. These two results mean that both neural networks start perceiving both corpora employed in the mixup procedure as efficiently as if the networks were trained on each data set separately. Due to its simpler architecture, the ComParE_func classifier does not suffer from the overfitting issue to such a high degree as the ANN-based models do (compare Figure 5.4 and 5.10).

A similar trend can be spotted in Figure 5.11 that demonstrates experiments on merging all three corpora: if trained on all the corpora without using Algorithm 2, both ComParE_LLD and E2E models slightly disregard SVC and completely neglect HB in favour of VACC. The latter corpus has the highest correlation between its features and labels and hence can be easily learned by our classifiers. The proposed Algorithm 2 allows us to train a multipurpose ANN that performs equally well on each of the corpora as if there were three networks trained exclusively for individual tasks. The classification performance obtained on VACC and HB with the ComParE_func model is generally lower compared to the results of the ANN-based models, and mixup is unable to improve it.

5.5 Experiments on the Restaurant Booking Corpus

In this section, we jointly study classical H-M AD (represented by VACC and SVC) and complexity-identical H-M AD (represented by RBC). We dedicated Section 5.4 to multitask learning and general mixup capabilities towards acoustic AD, while the present section is aimed at improving the classification performance on the most challenging of our corpora (RBC) by means of cross-corpus data augmentation and data fusion.

5.5.1 Cross-Corpus and Multitask Classification Using Simple Models

First, we conduct two series of KOCI and LOCO experiments with the ComParE_func classifier to estimate its behaviour on various corpora. The methodology of these experiments is similar to that one from Section 5.4.1. This time, we confine to the following corpora: VACC, SVC, and RBC. The classifier is trained and tested on the data partitions from Table 4.2. Within these experiments, we do not perform feature selection and do not use mixup. Results of both experimental series are presented in Figure 5.12. Let us denote the matrix from Figure 5.12a as $A'$, the matrix from Figure 5.12b as $B'$, and their elements as $a_{\text{train_corpus, test_corpus}}'$ and $b_{\text{corpus_excluded_from_train, test_corpus}}'$ respectively.
A Novel Approach to Cross-Corpus Data Augmentation Based on the Mixup Method

Fig. 5.11: Experiments with merging VACC, SVC, and HB simultaneously. The "+mix" mark means that Algorithm 2 is activated. The acoustic context window length being used by the ComParE_LLD and the E2E classifier equals 2 seconds.

\[ a'_{VACC,SVC} \] and \[ a'_{SVC,VACC} \] are considerably greater than the other off-diagonal elements of \( A' \), demonstrating a clear relationship between VACC and SVC. The other off-diagonal elements of \( A' \) are close to a random-choice UAR of 0.5, revealing no relationship between RBC and the other two corpora. However, \( a'_{RBC,VACC} \) is greater than \( a'_{RBC,RBC} \), meaning that the model trained on RBC performs better on VACC than on the RBC test set. In other words, training on the complexity-identical H-M AD setup allows the model to learn true acoustic addressee patterns that are also typical for the classical H-M AD setup. However, the opposite statement does not work due to collateral factors, such as differences between the human addressee’s appearance and the SDS appearance, between the domains and complexity levels of human-directed and machine-directed utterances, etc. These differences influence the user’s behaviour, particularly speech and prosody, within the VACC setup, but not within the RBC setup, and cause the model to overfit. SVC and RBC do not exhibit such a relationship probably due to the difficult acoustic conditions of SVC (it is the only corpus out of the considered three that contains not only indoor recordings but also outdoor recordings made with a distant microphone).

Another remarkable observation is the elements \( b'_{VACC,RBC} \) and \( b'_{SVC,RBC} \) being greater than \( a'_{RBC,RBC} \) and meaning that merging the acoustic data of RBC and the acoustic data of another corpus improves the classification performance on RBC compared to using the RBC training data alone. This positive result is similar to that one obtained for text (see Figure 4.20b). It motivates us to employ Algorithm 2 in Section 5.5.3 rather than just merging training examples from different corpora.
5.5 Experiments on the Restaurant Booking Corpus

5.5.2 Feature Selection

Second, we want to determine the degree of similarity between our corpora in the sense of common relevant acoustic features and apply RFE in combination with the ComParE_func model for this purpose. The coefficients of the normal vector of the linear SVM are used as attribute weights similarly to the experiments from Section 4.4.2. Figure 5.13 demonstrates RFE curves obtained by applying ten-fold LOSGO on various corpora. The resulting performance is calculated as UAR averaged over all folds for each reduced feature set. One set of curves visualises RFE on individual corpora. These curves are identical to those presented in Figure 4.11. Another set of curves marked with an asterisk reflects joint RFE across all three corpora. In order to obtain the latter curves, we compute a single weighting list on all the corpora merged in equal proportions and use a single ComParE_func model, which is also trained on all the corpora, to calculate the classification performance on them. We consider a feature set to be optimal if further RFE iterations worsen the classification performance stably.

Intersections of three optimal feature sets, each of which was obtained on a random fold with RFE on individual corpora, are presented in Figure 5.14. The representative acoustic functionals vary considerably: VACC, SVC, and RBC have only 1170, 2020, and 1570 relevant features out of 6373 respectively, while having only 139 attributes in common. However, these common functionals concern almost each of the 130 IS13_ComParE LLDs in some form. All three categories of the LLDs are involved (energy-, spectrum-, and voicing-related) alongside with the LLD deltas. The following LLDs are absent among the 139 common functionals:

- Energy-related, prosodic: audspecRasta lengthL1norm sma;
- Spectrum-related, spectral: pcm fftMag spectralHarmonicity sma;
- Voicing-related, voice quality: jitterLocal sma, logHNR sma.
Fig. 5.13: Acoustic feature selection using RFE on VACC, SVC, and RBC.

Fig. 5.14: Comparison of the optimal acoustic feature sets obtained with RFE on VACC, SVC, and RBC. These sets correspond to the curves from the first row of the legend (without an asterisk) in Figure 5.13.
5.5 Experiments on the Restaurant Booking Corpus

The IS13_ComParE feature notations can be found in [Eyben, 2015] and Section 2.2.1. We conclude that, in our case, the feature selection is basically reduced to the selection of the optimal functionals for each LLD contour. We apply the ComParE_LLD model in Section 5.5.3 for this purpose. Due to its recurrent architecture, this model can learn its own, "custom" functionals from the time-continuous LLD sequences for the particular task.

According to the joint RFE experiments across all three corpora (see the curves marked with an asterisk in Figure 5.13), the entire IS13_ComParE set of 6373 acoustic functionals should be used for joint classification since the classification performance on VACC and RBC starts dropping directly after the first RFE iterations. In particular, merging RBC with the other corpora demonstrates the greatest benefit for acoustic AD on RBC when the entire IS13_ComParE set is employed. In other words, the 139 common functionals visualised in Figure 5.14 are important but not enough to classify RBC utterances reliably.

5.5.3 Speech Data Augmentation of the Restaurant Booking Corpus

Both SVM-based and ANN-based classifiers show considerably lower performance values on RBC compared to the other two corpora. In contrast to Mallidi et al. [2018], C.-W. Huang et al. [2019], and Norouzian et al. [2019], we do not have an Amazon-scale amount of data for training our deep models, though data augmentation helps us to combat this issue. In the present section, we conduct a series of experiments, simply merging RBC with the other two corpora or doing the same in combination with Algorithm 2. Results of these experiments conducted on the data partitions from Table 4.2 are presented in terms of UAR and UAP in Table 5.2.

Taking into account the results reported in Section 4.4.3, we consider a single length of acoustic context windows for our experiments with the ComParE_LLD and the E2E classifier in the present section. This length equals 2 seconds.

We noted that applying several regularisation techniques, e.g., mixup, Gaussian noise, and high dropout ratios, at once eliminated their positive effect, and therefore we use small dropout ratios and no additional noise when Algorithm 2 is activated. The performance values obtained with the models (1) – (4) on the RBC development set demonstrate that these classifiers benefit from merging RBC and another corpus and especially from merge(RBC, VACC). According to the ten-fold LOSGO cross-validation and a $t$-test with a significance level of 0.05, these classification improvements are significant. However, merging all three corpora is not the most effective training strategy due to a small proportion of RBC examples in the augmented training data: sometimes the model starts neglecting RBC in favour of the other corpora. Algorithm 2 does not provide any significant performance improvement for the SVM-based models (1) and (2) on the RBC development set due to their simple architectures that do not require regularisation. Mix(RBC, VACC) and mix(RBC, SVC) significantly improve the performance of the ComParE_LLD model (3) on the RBC development set, while significantly worsen the performance of the E2E model (4) compared to merging the same corpora without mixup. The latter result confirms our conclusion drawn in Section 5.4.3: when applied to raw audio signal, Algorithm 2 simply overlaps two audio files and hence does not differ from noise augmentation essentially. Furthermore, the algorithm may cause the cocktail party effect confusing the model. Another possible explanation of this negative result is trivial: the E2E classifier requires a larger amount of real training data to benefit from Algorithm 2.
Table 5.2: Classification performance on RBC in terms of UAR (upper value) and UAP (lower value). Each of the columns (1) – (4) has two subcolumns. The left subcolumn contains performance values computed on the RBC development set, the right subcolumn contains performance values obtained on the RBC test set. The other performance values are computed on the RBC test set. A bold value denotes the best result in the column/subcolumn. The underlined bold value shows the best result overall. Classifiers: (1) – ASR_meta, (2) – ComParE_func, (3) – ComParE_LLD, (4) – E2E, (5) – AP_meta, (6) – baseline classifier, (7) – non-native listeners, (8) – native listeners. Configuration of the model (5): ASR_meta(merge(RBC, V ACC)) + ComParE_func(merge(RBC, V ACC)) + ComParE_LLD(mix(RBC, V ACC)) + E2E(merge(RBC, V ACC)). The classifiers (1) and (2) utilise entire utterances. The classifiers (3) and (4) use context windows of 2 seconds. The “merge” mark denotes a simple merger of several corpora. The “mix” mark denotes that Algorithm 2 is activated. The “+” mark denotes fusion using the AP_meta classifier depicted in Figure 5.2.

<table>
<thead>
<tr>
<th>Data augmentation method</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBC</td>
<td>.539 .550</td>
<td>.533 .541</td>
<td>.548 .556</td>
<td>.552 .570</td>
<td>.628</td>
<td>.539 .544</td>
<td>.596</td>
<td></td>
</tr>
<tr>
<td>mix(RBC)</td>
<td>.643 .659</td>
<td>.565 .589</td>
<td>.559 .572</td>
<td>.558 .578</td>
<td>.632</td>
<td>.540 .544</td>
<td>.591</td>
<td></td>
</tr>
<tr>
<td>merge(RBC, V ACC)</td>
<td>.677 .590</td>
<td>.579 .601</td>
<td>.584 .598</td>
<td>.591 .609</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mix(RBC, V ACC)</td>
<td>.617 .630</td>
<td>.619 .638</td>
<td>.599 .611</td>
<td>.606 .622</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>merge(RBC, SVC)</td>
<td>.574 .588</td>
<td>.577 .598</td>
<td>.604 .620</td>
<td>.548 .569</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mix(RBC, SVC)</td>
<td>.615 .628</td>
<td>.616 .635</td>
<td>.613 .631</td>
<td>.567 .581</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>merge(RBC, V ACC, SVC)</td>
<td>.546 .567</td>
<td>.549 .570</td>
<td>.547 .567</td>
<td>.563 .587</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mix(RBC, V ACC, SVC)</td>
<td>.585 .598</td>
<td>.580 .599</td>
<td>.556 .577</td>
<td>.575 .598</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>merge(RBC, SVC, V ACC)</td>
<td>.554 .569</td>
<td>.546 .569</td>
<td>.563 .582</td>
<td>.513 .529</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mix(RBC, SVC, V ACC)</td>
<td>.581 .597</td>
<td>.584 .597</td>
<td>.576 .591</td>
<td>.537 .554</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The aforementioned differences between the performance values of individual classifiers remain for the RBC test set. However, the performance values on this set are slightly higher than the corresponding ones on the RBC development set as the former values were obtained using the RBC development set in addition to the training sets.

The following configuration of the AP_meta model has been chosen for metafusion: ASR_meta(merge(RBC, V ACC)) + ComParE_func(merge(RBC, V ACC)) + ComParE_LLD(mix(RBC, V ACC)) + E2E(merge(RBC, V ACC)). The AP_meta classifier (5) demonstrates a UAR of 0.628 and a UAP
of 0.632 on the RBC test set. We compare these results with the AD performance of the baseline classifier (6) and with the non-native (7) and native listeners’ (8) classification performance, applying ten-fold LOSGO and a t-test with a significance level of 0.05. The proposed AP_meta classifier significantly surpasses the baseline classifier and both groups of the human listeners on RBC in terms of both UAR and UAP. Each individual classifier of the AP_meta model also significantly surpasses the baseline classifier and the non-native listeners and performs at the native listeners’ level. The baseline classifier and the perceptual experiments with the human listeners are described in [Siegert et al., 2019], Section 3.3.5 and 3.3.6.

5.6 Fusion of Speech and Text

We select the best performing audio classifiers for each corpus and fuse them with the Sem_BLSTM model depicted in Figure 4.4. This version of the Sem_BLSTM classifier is trained using utterance-level addressee labels and the ASR transcripts similarly to the version proposed in Chapter 4. Fusion results are given in Table 5.3. We note that the late fusion (in the far right column of Table 5.3)
improves the classification performance of individual modalities for each corpus. The performance values demonstrated on VACC and RBC should be recognised as the new competitive baselines for these corpora. Although we did not manage to surpass the performance of the best baseline classifier proposed by Schuller et al. [2017], our E2E model reaches a UAR of 0.640 that makes it the best performing individual model on HB. The end-to-end classifier from [Schuller et al., 2017] showed a UAR of just 0.609 on the original HB development set that is equivalent to the HB test set in our terminology. We achieved this performance improvement due to a more careful choice of the CNN component architecture.

5.7 Summary

The classical H-M AD and the H-H AD problem have turned out to be essentially different in certain aspects. These aspects concern the acoustic and textual patterns of machine-directed and child-directed speech. On the one hand, neither the acoustic models nor the text classifiers managed to reveal any relationship between HB and the other corpora during our KOCI experiments presented in Figure 4.20a, 5.4a, 5.10a, and 5.10b. On the other hand, our LOCO experiments from Figure 5.4b, which use a linear acoustic classifier, demonstrate that the classical H-M AD and the H-H AD problem are non-contradictory.

The proposed Algorithm 2 is beneficial for DNNs employing handcrafted acoustic features, while not giving any significant performance improvement for end-to-end models which use raw speech data (see Section 5.4.3 for more detail). Linear classifiers do not benefit from Algorithm 2 neither due to their simple architectures that do not need to be regularised.

Another remarkable capability of the proposed Algorithm 2 was revealed in our multitask experiments presented in Section 5.4.4 and concerns both handcrafted acoustic features and raw speech data. The algorithm allows us to "mix" several corpora, which model different (but similar) classification tasks, in such a way that a single DNN trained on this mixture solves all the tasks equally efficiently as if there were several DNNs, each of which was trained on its own corpus exclusively. The data sets being employed for multitask learning with Algorithm 2 may essentially differ, e.g., VACC and SVC were uttered in German and collected for H-M AD in completely different domains. Moreover, HB was recorded for a different task (H-H AD) and uttered in a different language (English). Without Algorithm 2, the DNN emphasises the easiest corpus (with the strongest correlation between its features and labels, i.e., VACC), while neglecting the other corpora. Simple linear models do not suffer from this issue to such a high degree as DNN-based models do. However, the linear models demonstrate a lower classification performance overall.

Besides a clear relationship between VACC and SVC, which was captured by all of our acoustic classifiers in Figure 5.4a, 5.10a, and 5.10b, applying Algorithm 2 to these two corpora significantly improves the classification performance of the ComParE_LLD model on VACC at a context window length of two seconds (see Figure 5.7 for more detail).

Our experiments have also revealed a pragmatic difference between the classical and the complexity-identical H-M AD scenario. On the one hand, the necessity to choose between a human and a machine interlocutor within the same conversation appears to be a strong motivation for users to change their manner of speech in order to emphasise the desirable addressee. On the other hand, users demonstrate ambiguous addressee patterns in the absence of collateral factors, such as
the effect of a visible counterpart, different dialogue roles of addressees, different lexical content, different dialogue domains, and different complexity levels of human-directed and machine-directed utterances.

H-M AD in complexity-identical scenarios turned out to be a significantly more challenging problem than classical H-M AD. Acoustic and textual addressee patterns that take place in classical H-M AD also include those exhibited in complexity-identical H-M AD but are not only confined to them (see Figure 4.20a and 5.12a for more detail). Even native listeners can hardly resolve the addressee ambiguity in complexity-identical scenarios. Non-native listeners completely fail to classify the addressees in RBC so that any of our proposed models outperforms or at least keeps up with them. Employing the AP_meta classifier jointly with Algorithm 2, we managed to surpass the native listeners’ performance on RBC. The use of the out-of-domain data from other corpora together with RBC is highly beneficial, although the other two corpora were designed for classical H-M AD scenarios. See Section 5.5.3 for more detail.

Fusing our best performing models for speech and text analysis, we managed to improve the classification performance on each corpus, compared to the individual modalities and also to the existing baselines. See Section 5.6 for more detail on the updated competitive baselines.

The core methods and results reported in the present chapter have been published in [Akhtiamov, Siegert, et al., 2019; Akhtiamov et al., 2020; Kommersant, 2020].
Modern SDSs are getting more complex and human-like, obtaining more advanced communicational and perceptive abilities. An effective SDS is characterised not only by the set of different tasks it can solve but also by the system adaptability to users’ variable behaviour. Due to this quality of the SDS, users easily get engaged in the interaction process with the system and also express a higher loyalty level in favour of goods and services represented by the system. Moreover, the SDS can adapt its behaviour to users’ emotions, choosing appropriate phrases to calm down the user or considering negative emotions as a sign of some misunderstanding that may occur due to a system failure [Spirina et al., 2017]. Another solution that improves the SDS adaptability is H-M AD; the system learns to determine whether it is being addressed by the user or not. This ability especially comes in handy in multiparty conversations which may involve several humans talking to each other while interacting with the SDS (see Section 1.2.1 for more detail on H-M AD). In such conversations, the system is supposed to respond to system-directed utterances (On-Talk) explicitly, while human-directed utterances (Off-Talk) should be either ignored or processed in a specific, implicit way without giving a direct response to the user [Batliner, Hacker, and Nöth, 2008].

How do human beings specify and determine their addressees in real life? We possess several senses which enable collecting the necessary multimodal information that we leverage to determine whether we are being addressed as soon as somebody starts talking to us. Comprehensive lists of H-M AD-related features are given in [Tsai et al., 2015] and Section 2.2. Although the addressee cues used by humans in a H-H conversation are generally more implicit than those of a H-M conversation, these groups of cues have many features in common. The most evident of them are visual, i.e., head orientation and gaze, since humans tend to look at the object or person they are talking to [Maglio et al., 2000; Skantze and Gustafson, 2009]. Other cues are acoustic-prosodic; they are particularly relevant to special addressees, e.g., SDSs, children, hard-of-hearing people, and humans suffering from psycho-neurological disorders. These cues are based on the observation that humans tend to adjust their manner of speech to such addressees, making it more rhythmical and generally easier to understand as soon as the speakers start talking to the addressees since the former ones do not perceive the latter ones as adequate conversational partners [Batliner, Hacker, and Nöth, 2008; Shriberg et al., 2012; Shriberg et al., 2013]. Other cues are based on the semantic content that can be extracted from verbal information [H. Lee et al., 2013; S. V. Ravuri and Stolcke, 2014; S. Ravuri and Stolcke, 2015]. Humans may therefore specify desirable addressees explicitly by name as well as
implicitly relying on syntax or some contextual information, e.g., on the dialogue history, different roles of the interlocutors, etc. [Ouchi and Tsuboi, 2016; R. Zhang et al., 2018].

The present chapter is dedicated to multimodal H-M AD on SVC. This corpus, described in [Batliner et al., 2007] and Section 3.1, consists of three modalities: speech, manual transcripts, and video. The key advantage of SVC over the other corpora considered in the present thesis (see Section 3.2 – 3.4 for more detail on the other corpora) is the presence of the visual modality that makes video-based H-M AD possible. Furthermore, SVC is the only corpus containing the manual transcripts that enable trustworthy text analysis. In other words, these transcripts allow us to estimate the real contribution of the textual modality to H-M AD regardless of the ASR accuracy. In this light, we propose a novel approach to modelling acoustic-prosodic, textual, and visual aspects of classical H-M AD. We study each aspect individually and also jointly with the others, performing multimodal fusion at the decision level. In our case, the textual modality turns out to be the most informative for H-M AD, and therefore we especially focus on modelling its more detailed aspects, such as lexical content, structure of sentences (syntax), and semantic content. We also show how the AD performance of the proposed models depends on the degree of speech spontaneity. Finally, we introduce a new competitive baseline for SVC.

The present chapter is organised as follows. In Section 6.1, we propose several classifiers for individual modalities and also introduce a multimodal classifier capable of analysing video, audio, and text jointly. In Section 6.2, 6.3, and 6.4, we conduct experiments on the video, audio, and text data of SVC respectively. Finally, we report our experiments on multimodal H-M AD in Section 6.5. The content of the present chapter is summarised in Section 6.6.

### 6.1 Proposed Classifiers

In the present section, we propose several individual models for visual, acoustic, and textual H-M AD. We also introduce an approach to their fusion at the decision level.

#### 6.1.1 Frontal Face Detector

The first classifier (denoted as Frontal-Face-Viola-Jones or FF_VJ) employs several features representing head orientation. Eye tracking would also provide some additional information, e.g., gaze direction that could be useful for the H-M AD problem [Tsai et al., 2015]. However, a low video resolution of 320 × 240 pixels does not allow us to extract any reliable information regarding the pupil position, though leveraging this data would be reasonable under the conditions of a short-distance human-smartphone interaction that takes place in SVC.

For On/Off-View classification, it is sufficient to distinguish frontal faces from others. One idea would be to train an image classifier directly on On/Off-View labels by using the SVC video data as the authors of the corpus did in their original study [Noth et al., 2007; Batliner, Hacker, and Nöth, 2008]. Another idea is to apply a pretrained model for frontal face detection. Face detection is a common task that has many real applications. There also exist highly effective, flexible solutions which can easily be integrated into end systems for face detection. We have chosen the second option: we apply the OpenCV library and its off-the-shelf frontal face detector based on the Viola-Jones (VJ) method for image classification [Viola and Jones, 2004]. The main advantages of the VJ
method are low resource consumption, short computational time, and a small amount of the training data required. These features made it one of the most popular algorithms for object detection.

In the VJ method, Haar-like features are calculated, then an integral image (also known as a summed area table) is created. After that, a variant of the AdaBoost learning algorithm is applied to both select the best features and to train the classifiers that use them. As a result, we obtain a boosted cascade of weak classifiers that is ready for object detection. The predictions returned by the frontal face detector are obtained analysing video frames extracted from utterances at an interval of 0.1 seconds. These output values are directly considered as On/Off-View predictions.

Despite the simplicity of the proposed FF_VJ classifier, it is sufficient to classify the SVC video data; as we will see in the present chapter below, the visual modality plays an auxiliary role in multi-modal H-M AD. In other words, a small improvement of the On/Off-View classification does not influence the performance of the On/Off-Talk classification. Furthermore, the use of a more advanced image classifier does not provide any benefits due to a low quality and a small amount of the SVC video data. Besides the FF_VJ classifier, we also implemented the following two models for frontal face detection: an image classifier based on facial landmark detection and a two-dimensional CNN. Inside the first model, the following features were extracted at the frame level: number of recognised faces, number of recognised eyes on the largest face, area size of the largest face, distance between the centres of the eyes on the largest face, and distance between the eye level and the mouth level on the largest face. In theory, the latter three attributes allow us to track the distance between the user and the system, head rotations side to side, and head rotations up and down respectively. All the extracted features were fed to a linear SVM that returned a frame-level On/Off-View prediction. However, the proposed idea does not improve the On/Off-View classification mostly due to a low video resolution and poor lighting conditions leading to errors in detecting facial landmarks. The second model was a CNN receiving a two-dimensional array of greyscale pixels as input and returning a frame-level On/Off-View prediction as output. This model does not improve the On/Off-View classification neither due to a small amount of the training data available.

6.1.2 Acoustic-Prosodic Classifiers

We use the following two acoustic-prosodic classifiers: ComParE_func and ASR_meta proposed in Section 4.1.1 and 4.1.3 respectively. However, the version of the ComParE_func classifier implemented in the present chapter has an important modification: the input features are now normalised for each speaker similarly to [Batliner, Hacker, and Nöth, 2008] so that our classifier and the baseline classifier reported in the aforementioned paper are put under the same conditions.

6.1.3 Text Classifiers

We consider AD to be an utterance classification problem. In the present section, we propose three models that carry out different types of text analysis. Designing our models, we assume that simple classifiers (Lex and Synt) and single-vector text representations for them can be trained on in-domain data exceptionally, while the usage of out-of-domain data is reasonable for training more complex architectures employing DNNs (Sem).

Most studies on text-based H-M AD consider various aspects of the textual modality separately. For instance, Batliner, Hacker, and Nöth [2008] model the syntax of users’ utterances by means of
POS \(n\)-grams. H. Lee et al. [2013] use a similar \(n\)-gram-based approach to model lexical content. More advanced studies, e.g., [Mallidi et al., 2018], consider semantic content for H-M AD. However, we want to model the aforementioned aspects of text jointly to compare their classification performance. We also believe that their fusion can improve the total AD performance.

**Lexical Classifier**

The lexical classifier (Lex) employing an \(n\)-gram LM using word tokens performs lexical analysis at the utterance level. The idea behind this model is that there exist some typical lexical units for each addressee class. The Lex classifier implements the processing stages listed at the beginning of Section 2.2.3. At the tokenisation stage, we cast the input text to a sequence of words separated with spaces. All auxiliary characters, e.g., punctuation marks and special symbols, are removed, and all capital letters are brought to lowercase. At the filtering stage, we apply stop-word filtering, stemming, or both linguistic dimensionality reduction techniques at once. At the term extraction stage, we extract \(n\)-grams of various lengths (from uni- to trigrams). Text preprocessing examples at the latter two stages are given in Formula 2.1 – 2.25. At the term weighting stage, we weight the obtained terms by using the seven techniques described in Formula 2.26 – 2.48. Except TF-IDF, these methods are supervised, i.e., they take into account statistical information about the class labels and therefore simplify the classification problem for our classifier. At the dimensionality reduction stage, which is optional, we apply the feature selection based on term weights, the FT method (see Formula 2.49 – 2.53 and Algorithm 1 for more detail), or PCA described in Section 2.3.1. The obtained feature vector is fed to a linear SVM, a k-NN, or a Rocchio classifier discussed in Section 2.4.1, 2.4.2, and 2.4.3 respectively.

**Syntactical Classifier**

The syntactical classifier (Synt) performs syntactical analysis at the utterance level. The idea behind this model is that machine-directed utterances tend to match some specific syntactical patterns. The main difference between the Synt and the Lex classifier is the tokeniser: the syntactical model deals with POS tags or dependency tokens instead of real words (see Figure 2.2 for more detail). We also suppose that the Synt model requires a longer context compared to the Lex classifier. We therefore consider \(n\)-grams of greater lengths (from uni- to pentagrams) for the Synt model.

There exist many configurations of the Lex and the Synt model with various combinations of the operators at each processing stage of the pipeline given at the beginning of Section 2.2.3. The optimal set of operators should thus be found for the particular task. Considering that pipeline, we have the following alternatives for each processing stage with their number given in parentheses:

1. Tokenisation (1): plain text;
2. Filtering (6): no filtering, stop-word filtering, stemming, stop-word filtering & stemming, POS tags, dependency tags;
3. Term Extraction (5): uni-, bi-, tri-, tetra-, pentagrams;
4. Term Weighting (7): IDF, GR, CW, TM2, RF, TRR, NTW;
5. Dimensionality Reduction (4): no reduction, feature selection based on term weights, FT, PCA;

These sets of alternatives give us \(1 \times 6 \times 5 \times 7 \times 4 \times 3 = 2520\) configurations in total.
6.1 Proposed Classifiers

Semantic Classifier

Originally, the SVC data includes word-level addressee labels. Due to utterance segmentation errors and the speakers’ natural behaviour, some utterances contain words with different labels within the same utterance. Therefore, a natural idea is to use sequences of word embeddings together with a recurrent model for sequence-to-sequence prediction. Such a combination allows us to improve the classification performance by means of tracking addressee label changes at the word level within each utterance.

We want to leverage the full potential of the textual modality for H-M AD on SVC. Therefore, the version of the semantic classifier (Sem) proposed in the present chapter differs from the previous version, which was used in Chapter 4 and 5, in several aspects listed below.

First, we employ the SVC manual transcripts instead of the text output of a real ASR to eliminate the influence of ASR errors on the H-M AD performance.

Second, we use the original word-level addressee labels since they are more accurate than the utterance-level addressee labels employed in Chapter 4 and 5. Therefore, the BLSTM layer returns not just the last member of the output sequence but the entire output sequence. The RNN component hence returns not a single utterance-level addressee prediction but a sequence of word-level addressee predictions.

Third, we apply only those ANN layers which support masking. However, they are not optimised for working with GPUs. In other words, we sacrifice the computational time in favour of a higher classification performance. A masking operator is placed directly before the recurrent layer so that the network ignores padded empty time steps.

Fourth, the postprocessing component is activated. It averages the word-level confidence scores over all words in the utterance and returns an utterance-level confidence score required for fusion with other classifiers functioning at the utterance level.

The aforementioned aspects are reflected in Figure 4.4. The neural network hyperparameters remain the same as specified in Section 4.1.4.

S. Ravuri and Stolcke [2015] note that recurrent models are more sensitive to unknown words that may appear in the test data than \( n \)-gram models. A possible explanation for this phenomenon is that such a word affects the whole subsequent context perception of the utterance in case of recurrent models, while in case of \( n \)-gram models the same unknown word influences only the neighbouring \( n \)-grams that include it. We faced this problem during our first experiments with the recurrent models on the SVC data that contains around 20\% of singleton words in its vocabulary. Using the large out-of-domain corpora (TIGER and WikiNER), we managed to reduce the number of unknown words in the test data, though they still affected the classification performance. Most of the words remaining unknown turned out to be either proper names, or numerals, or compounds. The latter category comprises 40\% of all the words remaining unknown and appears due to the German habit to stack several simple words into long and complex constructions. To find representations for unknown words, we propose the following solution: according to the POS tags defined for each word prior to the syntactical analysis, each proper name and numeral is replaced by a word from the respective category in the out-of-domain corpora so that this word is already familiar to our system. It is sufficient for our classifier to know when it faces a proper name or a numeral; it does not need to distinguish different proper names or different numerals to solve our classification problem as these
words are irrelevant to H-M AD. The name of the system is an exception since it is important for H-M AD if used as a trigger-word. However, no advanced text analysis is required to detect the trigger-word; having an accurate ASR is more important in this case. Furthermore, the SVC experimental setup did not require users to pronounce a trigger-word. For each word that remains unknown, we search for a match among the words contained in the training vocabulary, that covers the maximum number of characters starting from the end of the unknown word. This is reasonable since the last sub-word of a German compound usually denotes its generic feature. If such a match is found, we represent it as an individual word that is already known to the system and continue the loop until no more matches are found, e.g., all sub-words in the compound Fußball+welt+meisterschaft (Football World Championship). For those words that remain unknown at the end, we employ a word hashing technique that is based on the idea described by P.-S. Huang et al. [2013]. We represent the unknown words as vectors of character trigrams, find the in-vocabulary word nearest to each out-of-vocabulary word in this space and then associate its representation with this in-vocabulary word.

More advanced approaches to handling out-of-vocabulary words are based on learning representations of individual characters (character embeddings), mapping them into word-level representations with some recurrent model, and then concatenating these word-level representations with ordinary word embeddings, such as those obtained with GloVe. The character embeddings of a word thus contain some representative information about the word even if the word embedding vector is unknown. Such accurate techniques are required for semantics-sensitive applications where the meaning of a word should be represented as precisely as possible [Kalpakchi and Boye, 2019]. Such a high accuracy is unnecessary for the H-M AD problem, where semantic meanings of words reflect only one aspect of users’ behaviour out of many.

Given the data enhancements and model improvements introduced in the present chapter, we expect the current version of the Sem classifier to outperform the previous version, employed in Chapter 4 and 5, on SVC.

6.1.4 Multimodal Fusion

The fusion of video, speech, and text is performed at the utterance level since the acoustic and the text classifier need to take into account the context of a whole utterance. However, the FF_VJ model provides us with On/Off-View predictions at the frame level, and they should also be transformed into utterance-level ones. The following six utterance-level metafeatures are thus extracted from the frame-level On/Off-View predictions: number of frames, proportion of On-View frames, and this proportion for each quarter of the utterance individually [Batliner, Hacker, and Nöth, 2008]. The latter five features play the roles of On/Off-View confidence scores in different regions of each utterance.
Fig. 6.1: Multi-meta classifier analysing video, audio, and text. The detailed architectures of the individual classifiers are depicted in the following figures: ComParE_func – Figure 4.1, ASR_meta – Figure 4.3, Sem – Figure 4.4. The FF_VJ classifier is discussed in Section 6.1.1. The Lex and the Synt classifier are described in Section 6.1.3.

- **Video**
  - Preprocessing
  - Frame extraction
  - Frontal face detector
  - Metadatum extraction
  - Normalisation
  - SVM 1 (linear kernel)
  - SVM 2 (radial kernel)
  - SVM 3 (linear kernel)
  - SVM 4 (linear kernel)
  - SVM 5 (linear kernel)

- **Audio**
  - Preprocessing
  - LLD extraction
  - Functional extraction
  - Normalisation
  - SVM 2 (radial kernel)
  - SVM 3 (linear kernel)
  - SVM 4 (linear kernel)
  - SVM 5 (linear kernel)

- **Text**
  - Preprocessing
  - Lexical text extraction
  - Syntactical text extraction
  - Normalisation (standard score)
  - SVM 3 (linear kernel)
  - SVM 4 (linear kernel)
  - SVM 5 (linear kernel)

Input (utterance)
We perform data fusion by concatenating the video-based metafeatures with confidence scores of other classifiers. Prior to the concatenation, each of the video-based metafeatures is brought to zero mean and unit variance on the development set. The other features are already normalised by nature. The resulting feature vectors are fed to a metamodel (denoted as Multimodal-Meta or Multi_meta) based on a linear SVM. All the standalone models are individually trained on the training set, while the metamodel is trained on the development set. This is done to endow the metamodel with better generalisation abilities as the standalone classifiers return confidence scores close to 0 or 1 when they see the same training examples again. The resulting performance of each model is calculated as UAR on the test set. The entire multimodal system is depicted in Figure 6.1.

6.2 Experiments on the Video Data

In the present section, we conduct a series of experiments with the FF_VJ classifier alone. Therefore, only the output of the FF_VJ pipeline (video-based metafeatures) is fed to the metaclassifier (SVM 5) in Figure 6.1. The other classifiers are disabled. Results of the experimental series performed on the data partitions from Table 4.2 are given in Table 6.1. Here the SVC test set contains 37 speakers instead of 41 since 4 speakers have been excluded due to technical problems related to video recording [Noth et al., 2007; Batliner, Hacker, and Nöth, 2008]. For consistency, this reduced test set is used in all experiments conducted within the present chapter. The FF_VJ classifier is trained on the union of the training (48 speakers) and the development set (10 speakers) of SVC.

The FF_VJ classifier with the frontal face detector fine-tuned on SVC was used in [Noth et al., 2007; Batliner, Hacker, and Nöth, 2008]. As expected, this classifier reaches a higher On/Off-View classification performance on SVC compared to the FF_VJ classifier based on the off-the-shelf frontal face detector [Noth et al., 2007]. However, both classifiers perform equally on the On/Off-Talk classification problem of SVC [Batliner, Hacker, and Nöth, 2008], i.e., a small improvement of the On/Off-View classification does not result in any significant improvement of the On/Off-Talk classification. Therefore, we will use the FF_VJ classifier based on the off-the-shelf frontal face detector for our future experiments in the present section and Section 6.5.

Although we classify On-Talk versus Off-Talk, it is also possible to estimate class recall for individual categories of speech (NOT, SOT, POT, and ROT, see Section 3.1.2 for more detail on the class labels) to determine the most problematic of them for the FF_VJ classifier. On-Talk is the same as NOT, and Off-Talk consists of SOT, POT, and ROT. Therefore, we compute a rectangular matrix $2 \times 4$, the meaning of which is similar to a square confusion matrix, and then calculate the class recall for each column, i.e., for each category of speech. Figure 6.2 shows class recall

Table 6.1: Performance of the FF_VJ model in terms of UAR on two classification tasks of SVC: On/Off-View (frame-level score) and On/Off-Talk (utterance-level score). All performance values are computed on the test set.

<table>
<thead>
<tr>
<th>FF_VJ classifier version</th>
<th>On/Off-View</th>
<th>On/Off-Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on the off-the-shelf frontal face detector</td>
<td>0.820</td>
<td>0.705</td>
</tr>
<tr>
<td>Based on the fine-tuned frontal face detector</td>
<td>0.884</td>
<td>0.705</td>
</tr>
</tbody>
</table>
Fig. 6.2: Performance of the video-based classifier (FF_VJ) on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively.

values obtained with the FF_VJ classifier for various speech categories on the data partitions from Table 4.2. As before, the classifier is trained on the union of the training (48 speakers) and the development set (10 speakers) and tested on the test set (37 speakers) of SVC. ROT turns out to be the most problematic speech category for the visual classifier demonstrating nearly a random-choice performance on it. However, there exists a natural explanation for this: visual information is useless in this case since the users are looking at the system display while reading the information on it aloud to their partner, as if they were talking to the SDS indeed. The second most problematic speech category for the visual classifier is SOT. This may be explained by the observation that this class includes short self-talk phrases, e.g., swear words, bringing ambiguity to the On/Off-Talk classification since users usually stay visually focused on the system display in such Off-Talk situations, as if they were addressing the SDS indeed.

6.3 Experiments on the Audio Data

We perform acoustic analysis of the SVC data, using the following two classifiers individually: ComParE_func and ASR_meta. In addition to the corpus normalisation, which has already been applied to the input of the ComParE_func model in Chapter 4 and 5, we carry out speaker normalisation in the present chapter. We normalise the extracted IS13_ComParE functionals for each speaker individually, assuming that the characteristics of their speech are known to us in advance. This assumption represents the greatest benefit that can potentially be obtained by means of speaker adaptation [Batliner, Hacker, and Nöth, 2008].
Similarly to the experiments reported in Section 4.4.2, we carry out RFE in the present section in order to improve the classification performance by removing irrelevant acoustic functionals that bring noise to our data. According to these experiments, the speaker normalisation influences the RFE results; the number of the IS13_ComParE functionals relevant to the SVC classification problem equals 2020 when the speaker normalisation is disabled (see Figure 5.13) and 1000 otherwise. However, both reduced feature sets include the 139 functionals forming the set $S_{VACC} \cap S_{SVC} \cap S_{RBC}$ in Figure 5.14.

We train our audio classifiers on the SVC training set (48 speakers) and then test them on the development (10 speakers) and test (37 speakers) sets specified in Table 4.2. Results of these experiments are given in Table 6.2. The performance on the test set tends to be higher than the performance obtained on the development set since in the former case the model is trained on both training and development sets, while in the latter case the training set is used alone. The ComParE_func_SN classifier surpasses the other models on both development and test sets. Given the performance on the development set, we will use the ComParE_func_SN classifier combined with the reduced feature set of 1000 IS13_ComParE functionals for our future experiments with data fusion in Section 6.5. Despite its low performance, the ASR_meta classifier will also be used for this purpose as it enables leveraging the ASR metainformation in addition to the speech data.

Applying a ten-fold LOSGO cross-validation on SVC, we conduct another series of experiments with the aforementioned classifiers in order to determine whether the differences between their performance values in Table 6.2 are significant. Results of the experiments are visualised in Figure 6.3. According to a $t$-test with a significance level of 0.05, the average performance values of all three models differ significantly. The ComParE_func classifier significantly outperforms the ASR_meta model, while the speaker normalisation significantly improves the performance of the ComParE_func classifier even further.

Similarly to the experiments visualised in Figure 6.2, we analyse the performance of the audio classifiers on various categories of speech. The experimental results obtained with the aforementioned models on the data partitions from Table 4.2 are shown in Figure 6.4. The classifiers are trained on the union of the training (48 speakers) and the development set (10 speakers) and tested on the test set (37 speakers) of SVC. There exists a common trend for both audio classifiers; they perform worse on POT and particularly on ROT than on SOT.

Table 6.2: Performance of the audio classifiers on the SVC development and test partitions. All values are given in terms of UAR.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR_meta</td>
<td>0.651</td>
<td>0.657</td>
</tr>
<tr>
<td>ComParE_func</td>
<td>0.723</td>
<td>0.767</td>
</tr>
<tr>
<td>ComParE_func_SN</td>
<td><strong>0.775</strong></td>
<td><strong>0.800</strong></td>
</tr>
</tbody>
</table>
6.3 Experiments on the Audio Data

Fig. 6.3: Performance of the audio classifiers on SVC (cross-validation). The "SN" mark denotes speaker normalisation. RFE is activated for both ComParE_func and ComParE_func_SN classifiers.

Fig. 6.4: Performance of the audio classifiers on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively.
It is important to note that the speech spontaneity essentially differs for each category, i.e., SOT contains completely spontaneous speech, POT consists of speech with moderate constraints (the user retells the system output to another human in arbitrary form), and ROT comprises speech with strict constraints (the user reads the system output aloud). In other words, audio-based models misclassify constrained human-directed speech more frequently than spontaneous human-directed speech. This result can be explained by the observation that machine-directed speech (NOT) is usually non-spontaneous neither since humans tend to formulate their requests word-for-word in mind before addressing the SDS. Therefore, the artificial acoustic-prosodic patterns of the speech categories with the strictest constraints (ROT and NOT) may easily be confused with each other.

6.4 Experiments on the Text Data

In the present chapter, all of the proposed text classifiers are trained and tested using the manual transcripts. We noted in Section 6.1.3 that the Lex and the Synt classifier could be configured in a lot of different ways. We compare various configurations of the text classifiers in order to find the optimal ones for the particular H-M AD task.

6.4.1 Lexical Analysis

First, let us choose the optimal $n$ value for the $n$-gram-based Lex classifier. Conducting a ten-fold LOSGO cross-validation on the SVC data without the test set, we compute the classification performance of various term weighting methods combined with a linear SVM for different $n$ values and visualise the results in Figure 6.5. Each performance value is averaged over all folds. The unigram LM is sufficient for reliable H-M AD as further context extensions do not improve the classification performance, making the text representations more sparse and noisy. The supervised RF method outperforms the other term weighting techniques. However, its performance is just slightly higher than the performance of the unsupervised IDF method.

Second, let us compare the classification performance of the SVM-based classifier with the performance of other classification methods: k-NN and Rocchio. This time, all the configurations are combined with unigrams only. The experimental results obtained on the same ten-fold LOSGO cross-validation are shown in Figure 6.6. Each performance value is averaged over all folds. The configuration including RF and SVM outperforms the other alternatives. None of the considered approaches to dimensionality reduction can improve the classification performance of this configuration. The k-NN and the Rocchio classifier have turned out to be significantly more sensitive to the choice of their hyperparameters than the SVM is; this sensitivity complicates their effective application to the particular text classification problem. It is also interesting to note that Rocchio is the only classification method for which the use of FT brings benefits in most of the cases. Apparently, this model is more sensitive to the dimensionality curse than the others are.
Fig. 6.5: Performance of various configurations of the SVM-based Lex classifier for different \(n\)-gram LMs. The "+FT" mark means that the feature transformation method is activated.

Fig. 6.6: Performance of various configurations of the unigram Lex model for different classification methods. The "+FT" mark means that the feature transformation method is activated.
The Lex classifier configuration including the unigram LM, the RF term weighting method, and the linear SVM has been chosen for future experiments in the present section and Section 6.4.4. An RF weight reflects the importance of the term, i.e., terms with higher weights are considered to be more important in the context of the particular classification task. Using the RF weights, we can therefore determine the most important lexical units (with the highest weights) for each category in the context of SVC. These lexical units are given in Table 6.3. We note that the users tend to address the SDS more formally than the human companion. Talking to the system, they prefer problem-oriented questions in polite form and avoid using pronouns that result in the anaphora resolution problem. In contrast to the machine-directed utterances, the human-directed speech usually contains pronouns, particularly second-person ones, and small talk, e.g., colloquial words and interjections. Indirect speech and other words retelling the previously obtained information are typical for POT and ROT.

Table 6.3: Most important lexical units (with the highest RF weights) for each category in the context of SVC.

<table>
<thead>
<tr>
<th>On-Talk</th>
<th>Off-Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Question words:</strong> welche (which), wann (when), wer (who)...</td>
<td><strong>Pronouns:</strong> ihn (him), dich (you), er (he)...</td>
</tr>
<tr>
<td><strong>Polite requests:</strong> möchte (would like), gern (willingly), würde (would)...</td>
<td><strong>Second person:</strong> du (you), dir (to you), deine (your)...</td>
</tr>
<tr>
<td><strong>Approximate time and location:</strong> morgen (tomorrow), Nähe (nearby), Nacht (night)...</td>
<td><strong>Colloquial words:</strong> okay, na (well), nee (nope)...</td>
</tr>
<tr>
<td><strong>Indirect speech:</strong> gefragt (asked), sagt (says), gesagt (said), erzählt (told)...</td>
<td><strong>Interjections:</strong> ah, oh, hm...</td>
</tr>
<tr>
<td><strong>Retelling the previously obtained information:</strong> folgende (following), beträgt (is), gefunden (found)...</td>
<td></td>
</tr>
</tbody>
</table>

### 6.4.2 Syntactical Analysis

Next, let us choose the optimal configuration of the Synt classifier. Given our experience of choosing the optimal configuration of the Lex classifier, we consider a linear SVM as the only classification method for the Synt model since this method is less sensitive to the choice of its hyperparameters and it also demonstrates a higher performance on the text classification using lexical features than k-NN and Rocchio do (see Section 6.4.1 for more detail).

Similarly to the lexical analysis conducted in Section 6.4.1, we need to determine the optimal \( n \) value for the \( n \)-gram-based Synt classifier. Conducting a ten-fold LOSGO cross-validation on the SVC data without the test set, we compute the classification performance of various term weighting methods combined with the linear SVM for different \( n \) values. Each performance value is averaged over all folds. As shown in Figure 2.2, we use the following two types of syntactic tokens: POS tags and dependency tags. The experimental results obtained with the classifiers employing the POS and the dependency tokens are shown in Figure 6.7 and 6.8 respectively.
Fig. 6.7: Performance of various configurations of the Synt classifier combined with the POS tokens for different $n$-gram LMs. The "+FT" mark means that the feature transformation method is activated.

Fig. 6.8: Performance of various configurations of the Synt classifier combined with the dependency tokens for different $n$-gram LMs. The "+FT" mark means that the feature transformation method is activated.
The trigram LM is sufficient for reliable H-M AD in both cases as further context extensions do not improve the classification performance, making the text representations more sparse and noisy. The Synt classifier configuration including the dependency tokens, the trigram LM, the RF term weighting method, and the linear SVM outperforms the other alternatives. Despite the fact that this classifier does not deal with real words but with abstract structures of sentences instead, its classification performance is quite close to the Lex model dealing with real lexical units. We have therefore chosen the aforementioned configuration of the Synt classifier for future experiments in Section 6.4.4.

6.4.3 Semantic Analysis

The Lex and the Synt model deal with text vectors, while the Sem classifier analyses sequences of word vectors instead. These sequences can be considered as time-continuous input. Therefore, we wonder how early it is possible to predict the addressee within one utterance. The SVC utterance length distribution depicted in Figure 6.9 has a mode value of around 10 words. However, the trend illustrated in Figure 6.10 demonstrates that the main performance gain is reached with the context of just the first six words; further context extensions contribute much less to the H-M AD performance. This result obtained on the manual transcripts matches the result computed on the ASR text output in Section 4.5.2. The resulting performance values have been calculated on the data partitions from Table 4.2.

![Kernel density estimation](image)

Fig. 6.9: Kernel density estimation of the SVC utterance length distribution for the textual modality.
6.4 Experiments on the Text Data

6.4.4 Comparison of the Text Classifiers

Within the present section, we compare the text classifier configurations introduced in Section 6.4.1 – 6.4.3. We train our text classifiers on the SVC training set (48 speakers) and then test them on the development (10 speakers) and test (37 speakers) sets specified in Table 4.2. Results of these experiments are given in Table 6.4. Measuring the performance of the metaclassifiers on the development set does not make any sense as this set has already been used for training their metamodels based on the Multi_meta classifier depicted in Figure 6.1. The corresponding cells of Table 6.4 thus remain empty. For the other classifiers, the performance on the test set tends to be higher than the performance obtained on the development set since in the former case the model is trained on both training and development sets, while in the latter case the training set is used alone. The Sem_BLSTM classifier surpasses the other individual models on both development and test sets. Given the performance on the development set, we will use this classifier for our future experiments with data fusion in Section 6.5.

According to Table 6.4, the Synt + Lex + Sem_BLSTM model demonstrates a small improvement over the individual Sem_BLSTM classifier on the test set. In order to determine the significance of this difference and the differences between the performance values of the other models given in Table 6.4, we apply a ten-fold LOSGO cross-validation on SVC. Results of these experiments are shown in Figure 6.11. According to a \( t \)-test with a significance level of 0.05, each of the three Lex classifiers significantly surpasses the Synt model. Despite the fact that the stop-word
filtering (Lex_F) and the stemming (Lex_S) reduce the dimensionality of the text representation (dictionary size) by 11% and 20% on average compared to the Lex classifier respectively, the stop-word filtering significantly worsens the performance of the Lex classifier, while the stemming does not influence the performance significantly. This result can be explained by the observation described in Table 6.3; the stop-word filter has removed some auxiliary words, e.g., second-person pronouns, that play important roles for H-M AD in the SVC setting. The fusion of the Synt and the Lex classifier significantly outperforms both individual models included in the metaclassifier. The Sem_LSTM model performs at the level of the aforementioned metaclassifier, meaning that the recurrent semantic model takes at least the lexical content and the syntactic structure of sentences into account. The Sem_BLSTM classifier significantly surpasses the Sem_LSTM model. In other words, considering both the forward and the backward context is beneficial for text-based H-M AD. However, the fusion of the Synt, the Lex, and the Sem_BLSTM classifier does not provide a significant performance improvement, meaning that we have exhausted the potential of the textual modality for H-M AD.

Similarly to the experiments visualised in Figure 6.2 and 6.4, we analyse the performance of the text classifiers on various categories of speech. The experimental results obtained with the aforementioned models on the data partitions from Table 4.2 are shown in Figure 6.12. The classifiers are trained on the union of the training (48 speakers) and the development set (10 speakers) and tested on the test set (37 speakers) of SVC. First, the Sem.BLSTM classifier provides a noticeable performance improvement for each category of speech compared to the other models. Second, there exists a common trend for all the text classifiers; they perform worse on POT and particularly on SOT than on ROT.
Fig. 6.11: Performance of the text classifiers on SVC (cross-validation). The "+" mark means fusion using the Multi_meta classifier depicted in Figure 6.1.

Fig. 6.12: Performance of the text classifiers on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively.
It is important to note that the speech spontaneity essentially differs for each category, i.e., SOT contains completely spontaneous speech, POT consists of speech with moderate constraints (the user retells the system output to another human in arbitrary form), and ROT comprises speech with strict constraints (the user reads the system output aloud). In other words, text-based models misclassify spontaneous human-directed speech more frequently than constrained human-directed speech. This result can be explained by the observation that SOT possesses a greater verbal variety than POT and ROT do. Therefore, SOT is more difficult to model and may be confused with machine-directed speech (NOT) sometimes. The behaviour of the text classifiers in Figure 6.12 is thus completely opposite to the trend exhibited by the audio classifiers in Figure 6.4.

### 6.5 Experiments with Several Modalities

Within the present section, we compare the best performing unimodal classifiers introduced in Section 6.2 – 6.4. We also fuse the individual models to obtain a multimodal classifier analysing visual, acoustic-prosodic, and textual information for H-M AD. We train our classifiers on the SVC training set (48 speakers) and then test them on the development (10 speakers) and test (37 speakers) sets specified in Table 4.2. Results of these experiments are given in Table 6.5. Measuring the performance of the metaclassifiers on the development set does not make any sense as this set has already

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual baseline (FF_VJ)</td>
<td>–</td>
<td>0.705</td>
</tr>
<tr>
<td>Acoustic baseline (prosodic n-grams (SN))</td>
<td>–</td>
<td>0.766</td>
</tr>
<tr>
<td>Textual baseline (POS n-grams)</td>
<td>–</td>
<td>0.760</td>
</tr>
<tr>
<td>Multimodal baseline 1 (prosodic n-grams (SN), POS n-grams)</td>
<td>–</td>
<td>0.808</td>
</tr>
<tr>
<td>Multimodal baseline 2 (prosodic n-grams (SN), POS n-grams, FF_VJ)</td>
<td>–</td>
<td>0.845</td>
</tr>
<tr>
<td>FF_VJ</td>
<td>0.703</td>
<td>0.705</td>
</tr>
<tr>
<td>ASR_meta</td>
<td>0.651</td>
<td>0.657</td>
</tr>
<tr>
<td>ComParE_func_SN</td>
<td>0.775</td>
<td>0.800</td>
</tr>
<tr>
<td>Sem_BLSTM</td>
<td>0.911</td>
<td>0.920</td>
</tr>
<tr>
<td>ASR_meta + ComParE_func_SN + Sem_BLSTM</td>
<td>–</td>
<td>0.924</td>
</tr>
<tr>
<td>ASR_meta + ComParE_func_SN + Sem_BLSTM (real ASR)</td>
<td>–</td>
<td>0.922</td>
</tr>
<tr>
<td>ASR_meta + ComParE_func_SN + Sem_BLSTM + FF_VJ</td>
<td>–</td>
<td><strong>0.927</strong></td>
</tr>
</tbody>
</table>

Table 6.5: Performance of the classifiers processing individual modalities and their fusion on the SVC development and test partitions. The upper part of the table contains baseline models from [Batliner, Hacker, and Nöth, 2008]. The lower part of the table contains the models proposed within the present thesis. All values are given in terms of UAR. The "SN" mark means speaker normalisation. The "+" mark denotes fusion using the Multi_meta classifier depicted in Figure 6.1.
been used for training their metamodels which are based on the Multi_meta classifier depicted in Figure 6.1. The corresponding cells of Table 6.5 thus remain empty. For the other classifiers, the performance on the test set tends to be higher than the performance obtained on the development set since in the former case the model is trained on both training and development sets, while in the latter case the training set is used alone. As expected, the fusion of the ASR_meta, ComParE_func_SN, and Sem_BLSTM classifiers outperforms the individual models included in the metaclassifier. This result represents the highest classification performance that can be achieved analysing speech only, i.e., its acoustic-prosodic information and verbal content. Introducing the visual modality, i.e., the FF_VJ model, into the aforementioned metaclassifier improves the performance on the test set even further. The proposed classifiers outperform the corresponding baselines from [Batliner, Hacker, and Nöth, 2008] for individual modalities and also for their fusion. We achieved such high results due to more advanced feature representations and more modern ML methodology compared to [Batliner, Hacker, and Nöth, 2008].

In order to determine the significance of the differences between the performance values of the classifiers given in Table 6.5, we apply a ten-fold LOSGO cross-validation on SVC. Results of these experiments are shown in Figure 6.13. According to a $t$-test with a significance level of 0.05, the performance values of the four unimodal classifiers differ significantly. The textual modality represented by the Sem_BLSTM classifier provides the greatest contribution to H-M AD in comparison with the other modalities. The contribution of the acoustic modality represented by

![Box plot](image-url)
the ComParE_func and ASR_meta classifiers is smaller compared to the contribution of text. The visual modality represented by the FF_VJ classifier makes the smallest contribution to H-M AD. The fusion of audio and text (ASR_meta + ComParE_func_SN + Sem_BLSTM) significantly improves the performance of the individual models included in the metaclassifier. We also examine the latter metamodel in combination with an off-the-shelf ASR [Google LLC 2019] (which has a word recognition accuracy of around 90% for German) to check the H-M AD performance under real conditions. As expected, the text data gets noisier (the vocabulary size is increased by 24%). However, we obtain an insignificant performance drop after replacing the manual transcripts by the ASR text output. Introducing the FF_VJ classifier into the metafusion alongside with the ASR_meta, ComParE_func_SN, and Sem_BLSTM classifiers leads to another significant performance improvement. This is the highest result that we have managed to achieve on SVC overall.

Similarly to the experiments visualised in Figure 6.2, 6.4, and 6.12, we analyse the classification performance before and after the multimodal data fusion on various categories of speech. The experimental results obtained with the best performing unimodal classifiers together with their fusion on the data partitions from Table 4.2 are shown in Figure 6.14. The classifiers are trained on the union of the training (48 speakers) and the development set (10 speakers) and tested on the test set.

![Figure 6.14](image-url)

**Fig. 6.14:** Performance of the best unimodal classifiers and their fusion on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively. "Fusion" means fusing the other four models by means of the Multi_meta classifier depicted in Figure 6.1.
(37 speakers) of SVC. As expected, the fusion of all four unimodal classifiers provides a higher performance for each category of speech than the individual classifiers do.

### 6.6 Summary

Within the present chapter, we have considered the following three modalities relevant to H-M AD on SVC: visual, acoustic, and textual. Text was shown to be the most important modality, the acoustic-prosodic data is the second most informative modality, while video plays an auxiliary role for H-M AD.

Considering the visual modality alone, we reduce the H-M AD problem to frontal face detection. However, this method is unable to distinguish ROT and also has difficulties when classifying SOT since users sometimes exhibit atypical head pose patterns, e.g., while swearing or reading something from the system display aloud. In other words, they keep looking at the display as a situational attractor, although they are talking to themselves or someone else. See Section 6.2 for more detail.

Considering the acoustic modality alone, we reduce the H-M AD problem to audio classification. This modality is represented by the following two components: ASR metafeatures and IS13\_ComParE functionals. The ComParE\_func model dealing with the latter component surpasses the ASR\_meta classifier processing the former component. Introducing the speaker normalisation into the ComParE\_func model improves the H-M AD performance even further. Both the ComParE\_func\_SN and the ASR\_meta classifier exhibit a similar behaviour on various categories of speech; they show a lower performance on constrained speech (POT and particularly ROT) than on spontaneous speech (SOT). See Section 6.3 for more detail.

Considering the textual modality alone, we reduce the H-M AD problem to text classification. This modality is represented by the following three components: lexical, syntactical, and semantical. The lexical component can be modelled by means of word unigrams, RF term weighting, and a linear SVM. Interestingly, stemming does not influence the performance of the Lex classifier for the German language, while stop-word filtering significantly reduces the classification performance, removing some important words, e.g., second-person pronouns. See Section 6.4.1 for more detail on the experiments with the Lex model. The syntactical component can be modelled with trigrams of dependency tokens, RF term weighting, and a linear SVM as well. Using POS tags instead of the dependency tokens is also possible. However, the dependency tokens provide a higher classification performance on SVC than the POS tags do. See Section 6.4.2 for more detail on the experiments with the Synt model. The semantical component can be modelled by means of GloVe word vectors and a BLSTM network. Using an LSTM network instead of BLSTM is possible as well. However, the BLSTM network considering both forward and backward context surpasses LSTM on SVC. Analysing the first six words of an utterance is sufficient for the Sem classifier to perform reliably on SVC. See Section 6.4.3 for more detail on the experiments with the Sem model. According to the results presented in Section 6.4.4, the Lex and the Synt model can be fully replaced by the Sem classifier that takes both the lexical content and syntactical structure of sentences into account. A fusion of all three aforementioned models does not improve the classification performance on SVC. Interestingly, the behaviour of the text classifiers on various speech categories is completely opposite compared to the audio classifiers; the text-based models show a lower performance on spontaneous...
speech (POT and particularly SOT) than on constrained speech (ROT). See Section 6.4.4 for more detail.

Considering video, audio, and text jointly, we fuse the best performing configurations of the classifiers processing individual modalities. As a result, we obtain a multimodal metaclassifier that significantly surpasses the individual classifiers on SVC. Furthermore, the multimodal system improves the classification performance on each category of speech compared to the individual classifiers. The multimodal system remains robust after we have replaced the manual transcripts by the text output of a real ASR. The proposed individual classifiers and their fusion outperform the existing baselines on SVC [Batliner, Hacker, and Nöth, 2008]. See Section 6.5 for more detail on the experiments with multimodal H-M AD.

The core methods and results reported in the present chapter have been published in [Sergienko et al., 2015; Akhtiamov et al., 2016; Akhtiamov, Sidorov, et al., 2017; Akhtiamov, Ubskii, et al., 2017; Pugachev et al., 2017; Akhtiamov and Palkov, 2018; Akhtiamov, Fedotov, et al., 2019].
Conclusions and Future Work

In the present chapter, we summarise the key results and contributions of our study. The work highlights are outlined in Section 7.1. The thesis contributions are listed in Section 7.2. We also propose some promising directions for future work in Section 7.3.

7.1 Overall Summary

We have studied three different problem statements of AD: classical H-M AD on VACC (desktop SDS) and SVC (portable SDS), complexity-identical H-M AD on RBC (call centre SDS), and H-H AD on HB (passively listening SDS). VACC and SVC have been collected for the same problem statement and have many things in common so that our acoustic and textual classifiers trained on one of them also demonstrate an adequate AD performance on the other corpus, despite the different SDS scenarios and domains of these two corpora. However, this is not the case for RBC and HB, although all the four corpora have a set of relevant acoustic spectrum-related features in common. Overall, our best performing acoustic and textual classifiers show UAR values slightly above 0.9 on VACC and slightly below 0.9 on SVC. Although SVC utterances have turned out to be a bit more difficult for our models to classify, leveraging video in addition to audio and text allows us to reach a UAR above 0.9 as well. With such a high AD performance, our systems can be employed in real-life applications to facilitate a more natural H-H-M interaction. It is also important to note that the approaches investigated within the present study do not require users to pronounce a trigger-word. However, the users sometimes called the system by name anyway. We therefore believe that the joint use of our paralinguistic analysers and a trigger-word recogniser can improve the AD performance of the proposed models even further. The presence of a trigger-word is beneficial for short utterances (up to six words) of VACC as this word usually appears at the beginning of some utterances. However, this benefit becomes insignificant for longer utterances.

Our experiments have revealed a pragmatic difference between the classical and the complexity-identical H-M AD scenario. On the one hand, the necessity to choose between a human and a machine interlocutor within the same conversation appears to be a strong motivation for users to change their manner of speech in order to emphasise the desirable addressee. On the other hand, users demonstrate ambiguous acoustic addressee patterns in the absence of collateral factors, such as the effect of a visible counterpart, different dialogue roles of addressees, different lexical content, different dialogue domains, and different complexity levels of human-directed and machine-directed
utterances. As a result, acoustic and verbal changes that take place in classical H-M AD include similar variations exhibited in complexity-identical H-M AD but are not only confined to them. This leads to extremely implicit addressee patterns and explains such a low AD performance obtained with our audio and text classifiers on RBC. However, the proposed models still outperform native German listeners, meaning that the problem of complexity-identical H-M AD remains challenging even for humans.

The problem of H-H AD differs from the H-M AD problems essentially. The AD performance obtained with the proposed audio and text classifiers on HB is significantly lower than their performance on VACC and SVC. The first reason for that is the poor quality of the HB recordings. Another reason is the fact that many HB utterances contain few words but many paralinguistic phenomena, e.g., whoops, shouts, laughter, etc. This results in multiple ASR errors and the absence of common acoustic and textual addressee patterns for child-directed utterances. A large number of child-directed utterances with zero recognised words also confirms the latter conclusion.

Despite the aforementioned differences between the analysed corpora, they also have some things in common. First, the context length sufficient for reliable AD equals two seconds or six words for all the corpora that matches the result obtained by Shriberg et al. [2013] for H-M AD in English. This context length corresponds to acoustic and textual addressee patterns at the utterance level. Further context extensions do not improve the AD performance achieved by the proposed classifiers. This result allows us to save computational resources when AD is carried out online. Second, the four corpora do not contradict each other, i.e., they can be learned jointly by a single classifier. Furthermore, a mix of SVC and VACC significantly improves the AD performance reached by the ComParE_LLD classifier on the latter corpus. Similarly, a mix of VACC and RBC significantly improves the AD performance achieved by the same classifier on the latter corpus. VACC and SVC effectively augment RBC in the sense of both speech and text data.

We have carried out a throughout multimodal analysis of the classical H-M AD problem on SVC. The corpus contains three modalities: audio, manual transcripts, and video. Each modality was represented by several feature sets. For the speech modality, we extracted a set of acoustic-prosodic functionals and a set of ASR metafeatures to model vocal addressee patterns. For the textual modality, we extracted lexical and syntactical n-grams of various context lengths and combined them with different linguistic filters and term weighting techniques to model the lexical content and syntactical structures of utterances. We also extracted semantic representations based on the GloVe word embeddings. For the video modality, we implemented the VJ method to distinguish frontal faces from the rest. If we place the designed feature sets in the order of their relative contribution to the total AD performance, starting from the most effective one, we will obtain the following sequence: semantic, lexical, syntactical, acoustic (functionals), visual, and ASR metafeatures. The following lexical units are the most relevant to H-M AD on SVC: question words, polite requests, approximate time and location, pronouns (especially second-person), colloquial words, interjections, indirect speech, and retelling the previously obtained information. The text classifiers perform worse on spontaneous human-directed speech than on constrained human-directed speech as spontaneous human-directed utterances contain a larger variety of lexical units which is more difficult to model. The audio classifiers show the opposite trend; they perform worse on constrained human-directed speech than on spontaneous human-directed speech as constrained human-directed utterances tend to have a more artificial prosody that can be easily confused with the prosody of machine-directed utterances. The
video classifier makes noticeable errors on spontaneous Off-Talk and completely fails on read Off-Talk as users are looking at the system while talking to other humans in such situations. Finally, we developed a multimodal classifier that includes the individual classifiers for each feature set. This model significantly surpasses the individual classifiers in terms of both the total AD performance and the AD performance on individual speech categories. The multimodal system remains robust after we have replaced the manual transcripts by the text output of a real ASR.

We have proposed several acoustic-prosodic and textual classifiers for AD. Three of them (ComParE_LLD, E2E, and Sem_BLSTM) are based on DNN architectures which have been fine-tuned for the given tasks. One of the three (E2E) is a fully end-to-end DNN capable of processing raw speech data. We have also proposed a novel multimodal classifier (Multi_meta) including several individual models for speech, text, and video analysis. Using our classifiers, we have managed to surpass the existing AD baselines on the analysed corpora.

Training our deep models for acoustic AD, we faced the problem of overfitting and proposed a novel approach to cross-corpus data augmentation that partially resolves it. Our mixup-based method not just improves the classification performance of a deep model which is trained and tested on a single corpus designed for one task. The proposed method also enables smooth multitask learning across various data sources by means of a single deep model. The corpora being employed for multitask learning may be collected for various (but similar) tasks and even uttered in different languages.

7.2 Thesis Contributions

The main contributions of the present thesis are divided into theoretical, practical, and experimental ones and listed in this section below.

7.2.1 Theoretical Contributions

- We have studied several special cases of AD for different SDS scenarios and domains, formulated their key differences and similarities, and also explained them in the sense of humans’ behaviour in real-life spoken applications.
- We have proposed a range of novel, effective classifiers, including DNN-based models, for acoustic and textual AD that outperform the existing baselines on the analysed corpora.
- We have proposed a novel approach to data augmentation that is based on the mixup method. The proposed approach not just provides effective regularisation for a deep model trained on a single corpus for a specific task; it also enables smooth multitask learning from several data sources by means of a single deep model.
- We have determined a set of features relevant to H-M AD for each data modality. These feature sets include acoustic-prosodic, lexical, syntactical, semantic, ASR metafeatures, and visual features. We have also proposed a novel multimodal classifier that analyses audio, text, and video and outperforms the existing baseline for SVC.
- We have demonstrated that the AD performance on audio and text depends on the degree of speech spontaneity. More specifically, acoustic-prosodic classifiers perform worse on con-
strained speech compared to spontaneous speech. Text classifiers demonstrate the opposite trend, showing a lower AD performance on spontaneous speech than on constrained speech.

### 7.2.2 Practical Contributions

- We have developed software for audio data augmentation.
- We have developed software for audio-based AD.
- We have developed software for text-based AD.
- We have integrated the developed software into a complex system for multimodal AD.

### 7.2.3 Experimental Contributions

- Using our classifiers, we have updated the existing baselines for VACC, SVC, RBC, and HB.
- We have determined some technical aspects which are relevant to AD, e.g., a set of informative acoustic features, the optimal length of acoustic context (two seconds), the optimal length of verbal context (six words), the most typical lexical units for each addressee category, etc. These experimental contributions can save computational resources when AD is carried out online.
- The presence of a trigger-word improves the AD performance for short utterances (below six words).
- We have studied the AD problems for another language besides English. Most of our results and conclusions, except those obtained on HB, are applicable to the German language.

### 7.3 Future Work

We have performed our AD experiments employing synthetic simulations, though the operational time and resources required for our classifiers, once we have trained them, match the capabilities of existing devices, e.g., smartphones or personal computers. This enables using our classifiers in real-life SDSs.

Paralinguistic information remains relevant to H-M AD for SDSs of the current generation. However, people will stop using paralinguistic signals in their machine-directed speech as the systems become more discerning and human-like. Users will no longer need to adapt their manner of speech to an artificial conversational partner that perceives and communicates just as they do. An H-H and an H-M conversation on the same topic will thus be indistinguishable in the paralinguistic sense. According to our experimental results, this trend can already be observed in the complexity-identical H-M AD setup of RBC. Therefore, the context and verbal content of users’ utterances will remain the only cues relevant to H-M AD. Furthermore, verbal information now plays the dominant role in multimodal H-M AD according to our experiments. In this light, the problem of H-M AD turns into a more challenging problem of addressee and response selection using the conversational context and verbal content of users’ utterances [Ouchi and Tsuboi, 2016; R. Zhang et al., 2018]. In some applications, visual cues, such as gaze direction, head position, or body posture, will also remain relevant to AD and can be integrated into multimodal systems as shown by Le Minh et al. [2018]. We consider this problem as the most promising future research direction for H-M AD.
It is also interesting to continue our experiments with the prosed approach to cross-corpus data augmentation based on the mixup method. In the present thesis, we tried two extremes: mixing handcrafted vectors of acoustic features and mixing raw audio signals. We believe that employing our method at intermediate levels, e.g., for mixing two-dimensional spectrograms or even for mixing custom feature vectors returned by a CNN-based feature extractor, would also be reasonable. We consider this idea as another promising direction for our future research.

Another interesting research direction is to simulate an H-H-M conversation by means of two groups of data sets. The first group would comprise only H-H conversations, the second one would consist of only H-M conversations. This artificial data mixture was shown to be effective for training acoustic and textual models on H-M AD [Shriberg et al., 2013; H. Lee et al., 2013]. Nowadays, this approach would be useful, particularly in combination with mixup, for training an end-to-end DNN together with its feature extractor for acoustic H-M AD as such models always suffer from a lack of training data.
References


Batliner, A., Hacker, C., Kaiser, M., Mögele, H., and Nöth, E. (2007). “Taking into account the user’s focus of attention with the help of audio-visual information: Towards less artificial human-machine communication”. In: AVSP.


McCulloch, W. S. and Pitts, W. (1943). “A logical calculus of the ideas immanent in nervous activity”. In: The bulletin of mathematical biophysics 5.4, pp. 115–133.


REFERENCES


REFERENCES 185


List of Own Publications

Journal Papers


Conference Papers


Press Releases

List of Figures

1.1 Pure H-M interaction involving a human being and an artificial conversational agent. .......... 3
1.2 Mixed H-H-M interaction involving H-M and Human-Human (H-H) types of interaction. This is the minimal representative example including two people and one artificial conversational agent... 3
1.3 Simplified H-H-M interaction. A user may talk either to a machine or to a companion who assists the user and does not talk to the system directly. ................................. 4
1.4 Example of a simplified H-H-M conversation. ....................................................... 4
1.5 Complexity-identical H-M AD. ................................................................. 5
1.6 Example of a conversation between two adults and a child. ............................... 6
1.7 Conceptual scheme of any SDS. ................................................................. 8
1.8 The place for an AD component in the SDS architecture. ..................................... 8
2.1 Experimental setup for realistic H-H-M interaction. Based on [Tsai et al., 2015]. ............. 20
2.2 Example of POS tagging and dependency parsing. ........................................... 29
2.3 Vector differences revealing a relationship between an adjective and its comparative and superlative forms. .................................................. 41
2.4 Two focuses of a user’s visual attention that were captured by a frontal camera placed above the display of a mobile SDS. ................................................ 42
2.5 Underlying idea of PCA. \( \{x_1, x_2\} \) is the original basis, \( \{x'_1, x'_2\} \) is a new basis (principal components). 45
2.6 Maximum-margin hyperplane. ....................................................................... 48
2.7 Example of a kernel function. ........................................................................... 51
2.8 Example of a simple k-NN-based classifier. The new data point depicted in black is assigned to the class of blue triangles if \( k = 2 \). The same data point is assigned to the class of green squares if \( k = 6 \). .................................................................................. 53
2.9 Example of a nearest centroid classifier (\( \gamma = 0 \)). The dashed geometric shapes denote the class centroids. The new data point depicted in black is assigned to the class of red circles since \( d_3 < d_2 < d_1 \). .............................................. 55
2.10 Multilayer fully connected ANN. .............................................................. 57
2.11 Architecture of an individual neuron \( AD_k \). .................................................. 57
2.12 Example of a CNN for classifying handwritten digits. Input image dimension and output dimensions for layer 1 – 4: \( (height \times width \times \#_\text{channels}) \). Output dimensions for layer 5 and 6: \( (\#_\text{neurons}) \). The red squares are filters. The light blue squares are channels. The number of the depicted channels is proportional but not equal to the number of the actual channels. .............. 62
2.13 Comparison of RNN and LSTM. Each line carries an entire vector from the output of one node to the inputs of the others. The circles represent elementwise operations. A split line means its content being copied and the copies going to different destinations. ....................... 64
2.14 Types of fusion. ......................................................................................... 67
List of Figures

3.1 H-H-M interaction represented in SVC ........................................ 71
3.2 H-H-M interaction represented in VACC ....................................... 76
3.3 Complexity-identical H-M AD problem represented in RBC .................. 80
3.4 H-H AD problem represented in HB ............................................. 84
4.1 ComParE_func classifier .......................................................... 91
4.2 ComParE_LLD classifier .......................................................... 93
4.3 ASR_meta classifier ............................................................... 94
4.4 Extended version of the Sem_BLSTM classifier. This version is applied in Chapter 6. The model version used in Chapter 4 and 5 differs from the depicted model in the following way. First, the mask is disabled. Second, the BLSTM layer returns only the last member of the output sequence. Third, the RNN component returns a single utterance-level prediction. Fourth, the postprocessing component is disabled .................................................. 96
4.5 Kernel density estimation of the VACC utterance length distribution for text ..................................................... 97
4.6 Kernel density estimation of the SVC utterance length distribution for text ..................................................... 97
4.7 Kernel density estimation of the RBC utterance length distribution for text ..................................................... 98
4.8 Kernel density estimation of the HB utterance length distribution for text ..................................................... 98
4.9 KOC1 experiments with acoustic AD on all four corpora. All results are presented in terms of UAR .......................... 103
4.10 Core idea of RFE using SVM weights ........................................ 104
4.11 Acoustic feature selection using RFE on all four corpora ..................... 105
4.12 Comparison of the optimal acoustic feature sets obtained with RFE on all four corpora. These sets correspond to the curves from Figure 4.11 ................................................. 106
4.13 Experiments with various acoustic context lengths on all four corpora. The performance curves are calculated using spline interpolation of exact measurements visualised as dots ............................................. 107
4.14 Kernel density estimation of the VACC utterance length distribution for audio ..................................................... 108
4.15 Kernel density estimation of the SVC utterance length distribution for audio ..................................................... 109
4.16 Kernel density estimation of the RBC utterance length distribution for audio ..................................................... 109
4.17 Kernel density estimation of the HB utterance length distribution for audio ..................................................... 110
4.18 Performance of the ASR_meta classifier on all four corpora ................ 112
4.19 Comparison of the vocabularies of the German corpora (VACC, SVC, and RBC) after stop-word filtering and stemming ................................................................. 113
4.20 Cross-corpus experiments with the Sem_BLSTM classifier on various corpora. All results are presented in terms of UAR ................................................................. 114
4.21 Experiments with various verbal context lengths on all four corpora. The word count starts from the beginning of each utterance ..................................................... 116
5.1 E2E classifier. The architecture of the RNN component is visualised in Figure 4.2 .......................... 121
5.2 AP_meta classifier including four low-level models. The detailed architectures of the ComParE_func, the ComParE_LLD, the E2E, and the ASR_meta classifier are given in Figure 4.1, 4.2, 5.1, and 4.3 respectively ................................................................. 123
5.3 Probability density function of the $\beta$-distribution with $\alpha = 0.5$ .................. 126
5.4 Preliminary cross-corpus experiments with the ComParE_func classifier on VACC, SVC, and HB. All results are presented in terms of UAR .......................... 128
5.5 Acoustic feature selection using RFE on VACC, SVC, and HB ................ 129
5.6 Comparison of the optimal acoustic feature sets obtained with RFE on VACC, SVC, and HB. These sets correspond to the curves from the first row of the legend (without an asterisk) in Figure 5.5 ................................................................. 130
5.7 Classification performance of the ComParE_LLD model combined with Algorithm 2 on VACC .......................... 131
5.8 Classification performance of the ComParE_LLD model combined with Algorithm 2 on SVC .... 132
5.9 Classification performance of the ComParE_LLD model combined with Algorithm 2 on HB. 132
5.10 Cross-corpus experiments with the ComParE_LLD and the E2E classifier on VACC, SVC, and HB. The "+mix" mark means that Algorithm 2 is activated. All results are presented in terms of UAR. The acoustic context window length being used equals 2 seconds. 134
5.11 Experiments with merging VACC, SVC, and HB simultaneously. The "+mix" mark means that Algorithm 2 is activated. The acoustic context window length being used by the ComParE_LLD and the E2E classifier equals 2 seconds. 136
5.12 Preliminary cross-corpus experiments with the ComParE_func classifier on VACC, SVC, and RBC. All results are presented in terms of UAR. 137
5.13 Acoustic feature selection using RFE on VACC, SVC, and RBC. 138
5.14 Comparison of the optimal acoustic feature sets obtained with RFE on VACC, SVC, and RBC. These sets correspond to the curves from the first row of the legend (without an asterisk) in Figure 5.13. 138

6.1 Multi_meta classifier analysing video, audio, and text. The detailed architectures of the individual classifiers are depicted in the following figures: ComParE_func - Figure 4.1. ASR_meta - Figure 4.3. Sem - Figure 4.4. The FF_VJ classifier is discussed in Section 6.1.1. The Lex and the Synt classifier are described in Section 6.1.3. 151
6.2 Performance of the video-based classifier (FF_VJ) on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively. 153
6.3 Performance of the audio classifiers on SVC (cross-validation). The "SN" mark denotes speaker normalisation. RFE is activated for both ComParE_func and ComParE_func_SN classifiers. 155
6.4 Performance of the audio classifiers on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively. 155
6.5 Performance of various configurations of the SVM-based Lex classifier for different n-gram LMs. The "+FT" mark means that the feature transformation method is activated. 157
6.6 Performance of various configurations of the unigram Lex model for different classification methods. The "+FT" mark means that the feature transformation method is activated. 157
6.7 Performance of various configurations of the Synt classifier combined with the POS tokens for different n-gram LMs. The "+FT" mark means that the feature transformation method is activated. 159
6.8 Performance of various configurations of the Synt classifier combined with the dependency tokens for different n-gram LMs. The "+FT" mark means that the feature transformation method is activated. 159
6.9 Kernel density estimation of the SVC utterance length distribution for the textual modality. 160
6.10 Performance of the Sem classifier on SVC (cross-validation). The word count starts from the beginning of each utterance. 161
6.11 Performance of the text classifiers on SVC (cross-validation). The "+" mark means fusion using the Multi_meta classifier depicted in Figure 6.1. 163
6.12 Performance of the text classifiers on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively. 163
6.13 Performance of the classifiers processing individual modalities and their fusion on SVC (cross-validation). The "+" mark means fusion using the Multi_meta classifier depicted in Figure 6.1. 165
6.14 Performance of the best unimodal classifiers and their fusion on various categories of speech. The approximate proportion of NOT, SOT, POT, and ROT utterances is 3:1:1:1 respectively. "Fusion" means fusing the other four models by means of the Multi_meta classifier depicted in Figure 6.1. 166
List of Tables

2.1 Co-occurrence probabilities of the target words *ice* and *steam* with some selected context words.  The statistics are extracted from a large textual corpus containing approximately six billion tokens.  
3.1 Example of ROT (italic). Original German utterances are given in a different font.  
3.2 Example of POT (italic). Original German utterances are given in a different font.  
3.3 Example of SOT (italic). Original German utterances are given in a different font.  
3.4 Label distribution as percentages of the total number of items in SVC.  
3.5 General characteristics of SVC.  
3.6 General characteristics of VACC.  
3.7 General characteristics of RBC.  
3.8 AD performance reached by human listeners and by an automatic classifier on RBC.  
3.9 HB partitions of the Interspeech 2017 ComParE Challenge.  
3.10 General characteristics of HB.  
4.1 Properties of the utterance length distributions visualised in Figure 4.5 – 4.8. All values are given in words.  
4.2 Train, development, and test partitions of the four corpora being considered.  
4.3 Properties of the utterance length distributions visualised in Figure 4.14 – 4.17. All values are given in seconds.  
5.1 Two-second UAR slices. Each marker corresponds to a curve of the same style in Figure 5.7 – 5.9. The greatest value in each row is marked in bold.  
5.2 Classification performance on RBC in terms of UAR (upper value) and UAP (lower value). Each of the columns (1) – (4) has two subcolumns. The left subcolumn contains performance values computed on the RBC development set, the right subcolumn contains performance values obtained on the RBC test set. The other performance values are computed on the RBC test set. A bold value denotes the best result in the column/subcolumn. The underlined bold value shows the best result overall. Classifiers: (1) – ASR_meta, (2) – ComParE_func, (3) – ComParE_LLD, (4) – E2E, (5) – AP_meta, (6) – baseline classifier, (7) – non-native listeners, (8) – native listeners. Configuration of the model (5): ASR_meta(merge(RBC, VACC)) + ComParE_func(merge(RBC, VACC)) + ComParE_LLD(mix(RBC, VACC)) + E2E(merge(RBC, VACC)). The classifiers (1) and (2) utilise entire utterances. The classifiers (3) and (4) use context windows of 2 seconds. The "merge" mark denotes a simple merger of several corpora. The "mix" mark denotes that Algorithm 2 is activated. The "+" mark denotes fusion using the AP_meta classifier depicted in Figure 5.2. 
5.3 Fusion of the best performing classifiers for speech and text on various corpora. The "merge" mark denotes a simple merger of several corpora. The "mix" mark denotes that Algorithm 2 is activated. The "+" mark denotes fusion using the AP_meta classifier depicted in Figure 5.2. "Score fusion" means fusing all classifiers from the same row at the decision level; their confidence scores are summarised and the label with the largest sum is chosen. "Metaclassifier" means introducing the text classifier (Figure 4.4) from the same row into the AP_meta model as a new low-level classifier.

6.1 Performance of the FF_VJ model in terms of UAR on two classification tasks of SVC: On/Off-View (frame-level score) and On/Off-Talk (utterance-level score). All performance values are computed on the test set.

6.2 Performance of the audio classifiers on the SVC development and test partitions. All values are given in terms of UAR.

6.3 Most important lexical units (with the highest RF weights) for each category in the context of SVC.

6.4 Performance of the text classifiers on the SVC development and test partitions. All values are given in terms of UAR. The "+" mark means fusion using the Multi_meta classifier depicted in Figure 6.1.

6.5 Performance of the classifiers processing individual modalities and their fusion on the SVC development and test partitions. The upper part of the table contains baseline models from Batliner, Hacker, and Nöth, 2008. The lower part of the table contains the models proposed within the present thesis. All values are given in terms of UAR. The "SN" mark means speaker normalisation. The "+" mark denotes fusion using the Multi_meta classifier depicted in Figure 6.1.